Multi-Robot Coordination in Domains with Intra-path Constraints

E. GIL JONES

CMU-RI-TR-09-31

THE ROBOTICS INSTITUTE
CARNEGIE MELLON UNIVERSITY
PITTSBURGH, PENNSYLVANIA 15213

August 2009

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

Thesis committee:

M. Bernardine Dias, Co-Chair
Anthony Stentz, Co-Chair
Paul Scerri
Lynne Parker, University of Tennessee, Knoxville

Carnegie Mellon

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

Thesis

Multi-robot Coordination in Domains with Intra-path Constraints

E. Gil Jones

Submitted in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the field of Robotics

ACCEPTED:

[illegible]		01/04/2010
M. Bernardine Dias	Thesis Committee Co-chair	Date
[illegible]		05/04/2010
Tony Stentz	Thesis Committee Co-chair	Date
[illegible]		May 10, 2010
Reid G. Simmons	Program Chair	Date

APPROVED:

[illegible]		5/24/2010
Randal E. Bryant	Dean	Date

Abstract

Many applications require teams of robots to cooperatively execute tasks. Among these domains are those in which successful coordination must respect intra-path constraints, which are constraints that occur on the paths of agents and affect route planning. One such domain is disaster response with intra-path constraints, a compelling application that is not well addressed by current coordination methods. In this domain a group of fire truck robots attempt to address a number of fires spread throughout a city in the wake of a large-scale disaster. The disaster has also caused many city roads to be blocked by impassable debris, which can be cleared by bulldozer robots. To produce coordinated agent behavior that satisfies the requirements of this domain entails not only allocating fires to fire trucks, but also determining what routes fire trucks should take given the presence of debris piles and which bulldozers should be assigned to clear debris along those routes. To determine high quality coordinated plans in domains with intra-path constraints requires that agent interactions be considered when making path planning, allocation, and scheduling decisions. This thesis addresses the problem of coordinating agents in such domains, with the goal of determining high quality coordinated behavior while minimizing the time required for computation.

Different scenarios involving domains with intra-path constraints will vary in terms of the nature of the constraints, the level of uncertainty in the environment, and the time available for computation. The central contribution of this thesis is a suite of different approaches for effectively coordinating behavior in domains with intra-path constraints, each of which has strengths and weakness and each of which may be most appropriate given scenario requirements and specifications. One technique we employ is greedy, market-based coordination. In our market-based approaches we seek to exploit the structure and independence inherent in domains with intra-path constraints to develop heuristics and bounding routines that serve to improve the quality of agent behavior while limiting the required computation. Our other primary technique is a randomized evolutionary search method that employs genetic algorithms. The genetic algorithm approach can potentially avoid local performance maxima that can result from using the market-based methods but requires orders of magnitude more computation.

In addition to our suite of approaches this thesis formulates domains with intra-path constraints as a distinct problem space within multi-robot literature and comprehensively treats the scenario factors that must be considered in approach design. Further contributions include a tiered auction methodology that extends market-based methods to domains where forming accurate bids requires soliciting the help of other agents; the first tractable time-extended allocation approach to domains with intra-path constraints; and the first approach that employs genetic algorithms within a domain where allocation and path-planning are inter-constrained.

Acknowledgments

Thanks to my committee for invaluable feedback on a number of levels. Special thanks to Tony and Bernardine, my advisors, for five years of sage counsel. I appreciate all the folks that have helped shape this thesis, including the members of the rCommerce lab and fellow Robotics Institute students Matt Zucker, Nathaniel Fairfield, and Brenna Argall among many others. I am also grateful for the love and support of my wife Loring and my family.

This work was sponsored by the U.S. Army Research Laboratory, under contract "Robotics Collaborative Technology Alliance" (contract number DAAD19-01-2-0012). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements of the the U.S. Government.

Contents

Glossary

ATC: allocate-the-coordinate
An approach that does not reason about explicitly scheduling coordination during task allocation, but after allocation uses route search to determine explicitly scheduled coordination solutions. 49

auction context
A value associated with a particular plan involving intra-path constraints that indicates that constraint-satisfaction assignments made during the auction should be considered provisional and incoporated only into context schedules. See also context schedule. 64

BFS: breadth-first search
A graph search approach that considers all neighboring nodes before considering descendents of those neighbors. 85

bidding function
A function which agents use to determine bids in a shared currency in order to participate in auctions. 62

coalition structure
The set of coalitions formed at a given time in domains that require the simultaneous efforts of multiple agents for addressing tasks. 156

constraint-addressing agent
An agent with the capability to address domain constraints. 6

context schedule
A copy of its current schedule that a constraint-addressing agent makes in order to bid on constraint-satisfaction duties associated with particular auction contexts. 65

coordination solution
A set of schedules for the agents in a team. 13

coordination space
The set of potential coordination solutions for a particular domain instance. 14

deadlock
A situation where two or more agents can make no further progress in executing their schedules due to indefinitely unsatisfied constraints. 146

Dispatcher/Auctioneer (D/A)
An agent who acts as a dispatcher for tasks and also as an auctioneer in our market-based approaches. 55

distributed approach
Computation that occurs on physically seperated computers and on local data without requiring information centralization. See also information centralization. 23

domain instance
A problem specification including a map of the environment, initial agent locations and characteristics, and task and constraint locations and characteristics. 14

dynamic domain instance
A domain instance where new information is learned during execution. See also domain instance. 15

explicitly scheduled coordination
Coordinated schedules that accurately account for inter-agent constraints in associating times with actions. 14

GA: genetic algorithm
An approach that uses randomized evolutionary methods for seach in the full coordination space. 52, 98

HOF: hall of fame
The hall of fame method for selecting a reproductive pool in the genetic algorithm. 104

HTN: hierarchical task network
A representation for planning that depicts tasks in terms of decompositions and inter-relations between tasks. 36

IA: instantaneous assignment
An approach that assigns each agent operating in a domain to at most one task at a time. 14

implicitly scheduled coordination
Coordinated schedules that do not accurately account for inter-agent constraints in associating times with actions. 14

independent coordination
A domain in which it is assumed that no inter-agent constraints exist that affect agents' ability to execute their schedules. 13

information centralization
The process of moving all potentially disparate state information from individual agents to a central location, where it can be reconciled into a global world view. 23

intra-path constraints
Constraints that occur on the paths of agents and affect route planning. 5

marginal reward
The extra reward an agent expects to achieve from accomplishing a task given that it has already committed to accomplishing other tasks. The marginal reward is computed by taking the difference between the summed reward for a schedule incorporating the new task and the summed reward for a schedule without the task. 56

MILP: Mixed integer linear program
A technique for optimizing a system of linear equations where some of the variables are required to be integers. 40

parallel computation
Computation that can be distributed onto different processors that each work in parallel on a single problem. 23

planning horizon
A horizon past which agent actions are not considered during the planning process. 14

POMDP: Partially observable Markov decision process
A Markov Decision Process in which agents cannot directly sense the underlying state of the world. 33

rendezvous
The process of achieving co-location for a set of agents. 156

scenario
A set of requirements and domain parameters that must be considered in approach design. 22

schedule
A series of actions an agent intends to take, the states it expects will result from those actions, and the times at which particular actions will be performed and states occupied. 13

solution quality metric
The primary objective function for judging the relative quality of behavior produced by agents acting in a domain. 14

static domain instance
A domain instance that is fully known and where new information is not learned during execution. See also domain instance. 15

TA-IA: tiered auctions with instantaneous task assignment
An approach that employs tiered auctions for instantaneous assignments. See also instantaneous assignment. 51

TA-TE: tiered auctions with time-extended task assignment
An approach that employs tiered auctions for time-extended assignment. See also time-extended assignment. 51

task-addressing agent
An agent with the capability to address domain tasks. 6

TE: time-extended assignment
An approach that potentially assigns more than one task to each agent operating in a domain. 14

terrainability
The ability of an agent to move through a location or area. 8

tight coordination
A domain in which agent schedules are constantly inter-constrained. 13

Chapter 1 Introduction

Autonomous mobile robots are playing an increasingly important role in many consumer, industrial, space, and military applications. Mobile robots with some measure of autonomy are vacuuming homes, making hospital deliveries, exploring Mars, and flying combat missions in Afghanistan and Iraq. Robots are most frequently employed in applications that are dangerous, dull, or dirty - these applications are generally unsuited for humans and are most appropriate for robotic solutions. Research efforts in robotics have increasingly turned towards enabling teams of robots to collectively address tasks rather than employing a single robot operating independently. A team of robots acting together can often outperform single robot solutions in terms of quality and robustness; at the same time, equipping multiple robots to cooperate to address tasks requires negotiating the substantial research challenge of coordinating the efforts of the robots.

This dissertation focuses on a type of multi-robot domain with a particular set of characteristics: domains with **intra-path constraints**. Intra-path constraints are constraints that occur on the paths of agents and affect route planning. For an example of such a domain, consider a disaster response domain that involves intra-path constraints. In disaster response a team of robots capable of extinguishing fires operates in a city that has been ravaged by a disaster of significant proportions. These fire truck agents must move to various locations around the city to extinguish fires. In addition to causing fires, the disaster has rendered many roads impassable; they are covered with debris and wreckage, creating obstacles that prevent fire extinguishing agents from navigating the environment. Suppose that there is another group of robots in the team that are designed to move freely in rough terrain, and can clear roads of wreckage and debris – these are bulldozer robots. An example of a problem instance in this domain is shown in Figure 1.1. Without the assistance of the bulldozers in clearing debris piles fire trucks would be incapable of reaching some fire locations; employing bulldozers as part of the team extends the ability of the team of fire trucks to cope with cluttered environments. At the same time, introducing another type of team member capable of

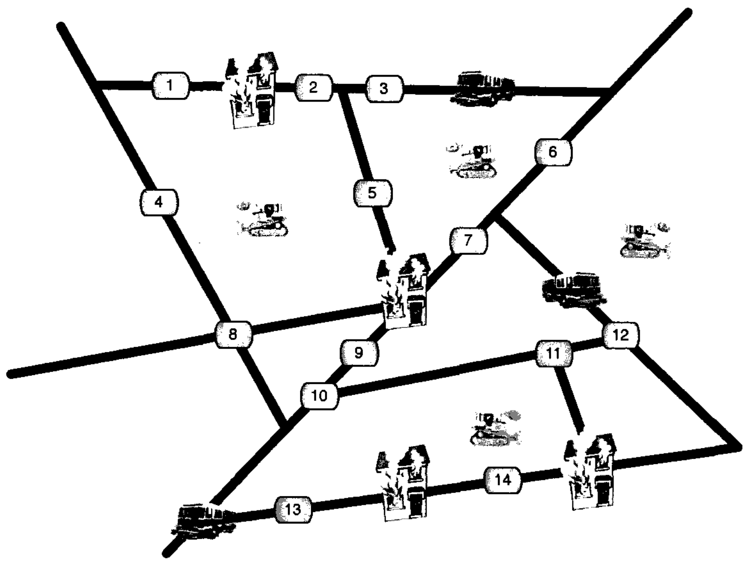


Figure 1.1: Example instance in a disaster response domain with intra-path constraints. Three fire trucks and four bulldozers are operating in a domain with four fires. Black lines represent roads, and ovals represent debris pile precedence constraints, labeled by number for reference.

assisting fire trucks makes the problem of coordination agent actions substantially more difficult; it now must be determined what assistance to provide, where to provide, and which agents should be involved.

The relationship between agents where one type of robot helps another move through the environment is a component of other domains beyond disaster response. For instance, minesweeper robots may need to be deployed ahead of time to allow autonomous ambulances to reach troop locations to evacuate injured soldiers, or aerial transport vehicles may be equipped to move ground-based science robots across difficult terrain like rivers or rubble. Intra-path constraints may also be present in domains where agents with different capabilities work together to acquire resources. In an industrial manufacturing domain, for instance, some mobile robots may have the capability to manufacture objects, but require the assistance of other agents in acquiring parts from inventories at different locations in a factory. In all of these domains some agents are equipped to address domain tasks - extinguishing fires, evacuating injured soldiers, conducting scientific exploration, or manufacturing items. However, these **task-addressing agents** lack some necessary capabilities to move through the environment and address domain tasks without the assistance of other agents. The assisting agents, **constraint-addressing agents**, are designed to perform particular functions at particular times and places to satisfy constraints and allow task-addressing agents to complete tasks. Furthermore, constraints affect the ability of task-addressing agents to either move through the environment or to acquire resources at particular locations; in either case, the presence of constraints and the dependence on constraint-addressing agents must factor into the routes that agents take through the space.

Domains with intra-path constraints represent a distinct category in multi-robot coordination that has not been fully addressed by the research community. The primary contribution of this dissertation is a suite of approaches for coordinating agents in domains with intra-path constraints. We present several distinct approaches to coordination for three primary reasons. First, different situations in which agents operate may have dramatically different resources available in terms of computing and available time; we present both approaches that use little computation and approaches that require substantially more computation for scenarios where resources are plentiful. Second, different domains may vary in terms of type and density of intra-path constraints, expectations of uncertainty, and number and kind of agents to be coordinated; the benefits of different solution styles can change depending on the features of the domain. Third, by formulating, evaluating, and comparing different styles of approaches in different situations we offer a better understanding of the problem of multi-robot coordination in domains with intra-path constraints.

Two primary themes present themselves in the progression of our suite of approaches. The first is that, generally speaking, searching more extensively in the space of possible ways to coordinate agents yields better performance in terms of domain requirements; however, searching more extensively also incurs greater computational cost. The second theme is that across all approaches we can potentially exploit particular features of domains with intra-path constraints to speed search and limit the computational resources required. In each solution approach we attempt to maximize the quality of coordinated behavior produced by the approach while minimizing the computational costs.

In the rest of this chapter we first more extensively describe different domains with intra-path constraints. We then introduce important concepts necessary for understanding the contributions of this thesis, examine the challenges of coordinating agents in domains with intra-path constraints, and offer insights into characteristics of the domain that can potentially be exploited to determine good coordination methods while using minimal of computation. This chapter concludes with a thesis statement and an outline of the dissertation.

1.1 Motivating domains with intra-path constraints

A number of compelling domains have intra-path constraints. In this section we introduce a few motivating examples of domains with intra-path constraints as well as discuss variations of each domain. We then discuss important unifying characteristics of domains with intra-path constraints.

1.1.1 Terrainability assistance

One substantial space of domains with intra-path constraints are those domains where some agents lack the ability to move freely through locations or areas, but can be provided with passage through sections of the environment by other robots. The ability to move in a given area is called having **terrainability** for that area, and in these domains constraint-addressing agents provide assistance by allow task-addressing agents to pass through area for which they lack terrainability. The disaster response domain with intra-path constraints is one such domain in this category. We first offer a greater level of detail about this domain, discuss variations of the disaster response domain, and then list other domains with terrainability assistance requirements.

Disaster response with intra-path precedence constraints

We have already introduced basic features of the disaster response domain, but have not discussed the methods by which we evaluate the performance of the team in the disaster response domain. We suppose the goal of the team is to preserve as much value as possible from affected buildings. When a fire occurs at a building the entire value of the building is put at risk - the building's value becomes the maximum reward offered for addressing the fire. Fires can be of different magnitudes, where the magnitude governs the rate of decay of the building value. The building value is exhausted when the decay causes the value to reach zero - no reward can be obtained from extinguishing the fire once the value reaches zero. By these assumptions the reward for extinguishing a particular fire is the affected building value less whatever damage has been caused by the fire in the interval before which the fire is extinguished. Agent behavior can be evaluated by assessing the sum of building value preserved by extinguishing fires at affected buildings. An example of this domain, with building values and decay rates, is shown in Figure 1.2.

Variations on disaster response

One potential source of variation in disaster response domains is in the interaction requirements at constraint sites. In some versions of the domain the efforts of a single bulldozer may be sufficient to clear any debris pile. In other domain versions, multiple bulldozers will need to work together to clear debris, representing simultaneity constraints between bulldozers at debris pile sites. In some domains fire trucks will need to collaborate to address fires, representing a simultaneity constraint at fire sites.

Another potential source of variation is in the expectation of uncertainty. Disaster response can involve static task issue, where the locations and characteristics of all fires are known up front and no new fires are discovered. If the expectation is that new fires will be constantly discovered at unpredictable locations then the domain exhibits dynamic task issue. It may be the case that the full set of intra-path locations are known up-front, or that new locations are dynamically discovered. Many other forms of uncertainty are also possible.

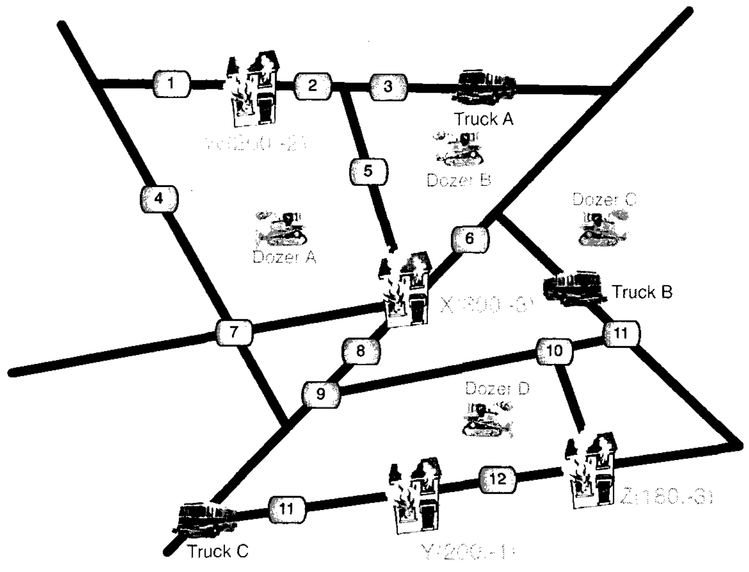


Figure 1.2: Example scenario in a disaster response domain with intra-path constraints. Three fire trucks and four bulldozers are operating in a domain with four fires. Orange fire labels indicate the current value associated with each fire – the first number in parentheses – as well as a rate of linear decay for the fire – the second number in parentheses.

Other domains with similar characteristics

Another domains with similar characteristics to disaster response is a mine-sweeping domain. In this domain mine-sweeping robots may be called upon to clear mines to allow the passage of medical robots moving to the locations of injured soldiers. Terrainability assistance relationships are also a component of a domain where robots capable of constructing temporary bridges or ramps work with scientific rovers to allow passage over chasms or craters on Mars or the Moon. In each of these domains there are pre-conditions along routes that must be met in order to allow passage to other agents.

Other domains may require constraints of simultaneity between agents seeking passage and those providing passage. In these domains agents must be co-located to provide the desired passage. One such domain could have crane robots capable of lifting other robots over barriers. A domain where some aerial robots could airlift ground robots over significant distances would also qualify, as the air-lifters and ground robots would need to co-locate to begin the air-lift.

1.1.2 Collaborative resource acquisition

Domains that focus on intra-path constraints preventing free movement of agents are only one class of domains with intra-path constraints. We also propose a set of domains where agents can move freely within the environment, but cannot necessarily address all requirements without planning paths that will allow constraint-satisfying interactions with other agents at particular times and places. Domains where agents must collaborate to acquire resources fall into this category. We again suggest one central motivating example, indicate variations, and offer other domains with these characteristics.

On-order manufacturing with intra-path precedence constraints

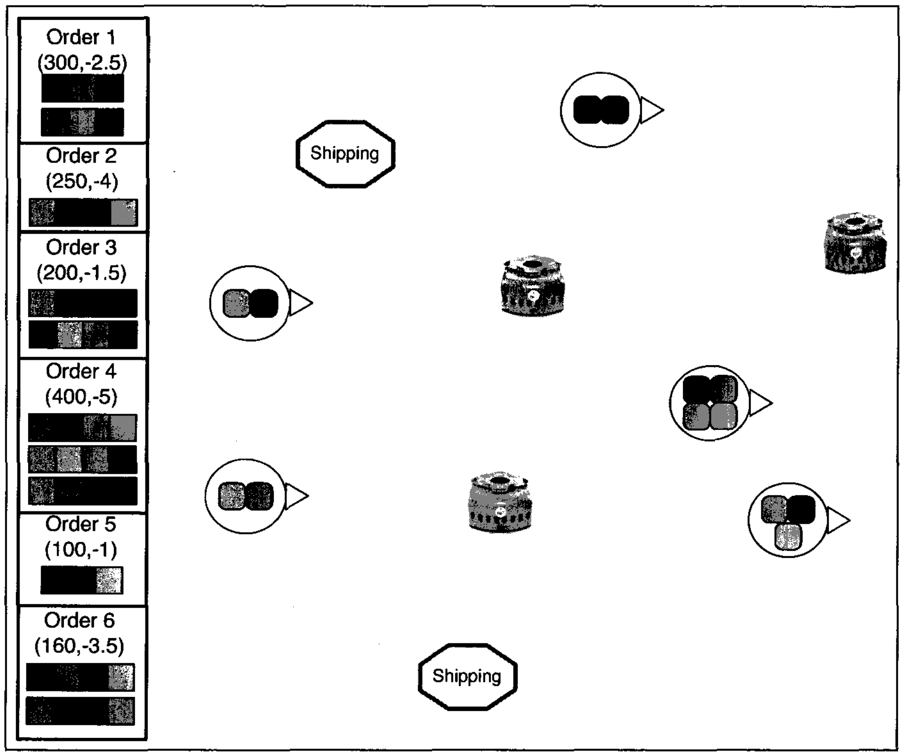


Figure 1.3: Example scenario in an on-order manufacturing domain. The current set of orders to be manufactured is shown at the left part of the figure, with the current reward and linear decay given in parentheses. The potential recipes for each order are shown in the order box, with colors representing the components that must be attached, and the sequence of components to attach in each recipe listed from left to right. The platform agents are the orange robots. The component inventory towers are shown as circles, with the components in each tower's inventory shown as colored boxes within the circles. The triangles attached to the platforms show the staging areas at which parts can be attached. Tower robots are not pictured.

For one example where agents need to work together to acquire resources, imagine a vast warehouse filled with towers of bins, each of which contains components to assemble a variety of different items. Orders for objects are passed into a dispatching system, with each ordering differing in importance and urgency. In order to fulfill the orders the objects must be manufactured. Each object has one or more recipes for object assembly, specifying a sequence of steps that need to be followed to successfully manufacture the item; assembly requires both acquiring the constituent parts and following the instructions for putting the parts together to create objects. Once created, objects must be delivered to one of a number of shipping areas. Two types of robots operate in the domain - one is a mobile assembly platform that can assemble items and move them to shipping locations. Another type of robot is attached to towers, and is capable of fetching objects and bringing them to staging areas at the base of towers for manufacture. Parts of the same type may be held in the inventory of several different towers, but not every tower will have every part. An example scenario for on-order manufacturing is shown in Figure 1.3, with the components required for recipes indicated by color. To determine high quality solutions mobile platform agents must determine what to manufacture, when to manufacture it, and which recipe to follow; they also must consider what route to take through the environment to acquire and attach the necessary parts and to deliver the parts to shipping areas. Tower agents must determine what to fetch to staging areas and when in order to satisfy intra-path precedence constraints.

Variations for on-order manufacturing

Potential variations could involve different kinds of intra-path constraints. Tower agents may fill a dual role of attaching objects to tasks, or there could be a third kind of agent in the staging area responsible for the fine manipulation tasks. In either case, the manipulating agent would need to co-locate with the platform agent to attach the part, representing a simultaneity constraint in addition to the precedence constraint of the tower agent having acquired the part.

There could be a variety of capacity constraints - staging areas could be limited to a certain capacity, and tower agents could potentially be able to carry more than a single item at a time. Platform agents could potentially have capacity to manufacture a single item at a time, or could have capacity to carry several items simultaneously.

Domains could have different expectations for uncertainty. Orders could either be static or dynamic. There could be uncertainty in part inventories, or durational uncertainty in the time required to fetch or attach parts.

Other domains with collaborative resource acquisition

On-demand harvesting of fruits, vegetables, or flowers is another potential area requiring resource collaboration, as it likely that the agents that will be harvesting the items will have a very different capability set from those transporting produce or assembling packages. In construction domains different agents may have different resources of abilities that need to be performed in particular sequences at particular locations, making this a potentially similar application area.

1.1.3 Common characteristics

Our examples of two classes of domains with intra-path constraints have some substantial differences but share a good deal of commonality. In each domain there is some set of tasks to complete with an associated time-sensitive value function which means that completing task requirements quickly is important. Each domain involves dividing task responsibilities between agents. In each domain there are task-addressing agents that must move in the domain environment in order to address domain tasks - the fire trucks in the disaster response example and the platform agents in the on-order manufacturing example. Each domain also contains constraint-addressing agents capable of assisting the task-addressing agents.

Dividing capabilities between task- and constraint-addressing agents is one possible division of labor, but other organizational principles could be employed. For instance, Okamoto et al. explore a vertical specialization strategy for multi-agent teams, where more plentiful, less capable agents occupy the bottom levels of a hierarchy (Okamoto *et al.*, 2008). However, from a system design standpoint we argue that a division between task- and constraint-addressing agents is a reasonable one. In the literature on multi-robot systems many different robots have been endowed with the abilities to address domain tasks. To extend the abilities of the team into new and more difficult arenas a natural step would be to develop robots that can provide assistance to the task-addressing agents. In the on-order manufacturing domain by putting inventory in parts towers a designer can extend the space of items that can be manufactured in a given area considerably. Similarly, an existing team of fire truck robots may be capable of operating in an environment with clutter-free streets, but totally stymied in an environment where streets were blocked by debris piles. While one option would be to make the fire truck agents more capable in terms of terrainability, such re-engineering may be difficult in many cases. For example, while making the fire trucks operate on tracks instead of wheels could give them terrainability in rough terrain, doing so could slow them down when operating on uncluttered streets. An easier approach may be to design a bulldozer robot specifically for the purposes of assisting the fire truck robots, extending the abilities of the team and enabling the design of more specialized robots. While extending our work to teams with arbitrary organizational structures is a potential area for future work, the task- and constraint-addressing framework is a useful one for structuring our approaches.

Another unifying feature of the domain areas is that the quality of a task-addressing agents' plan for a particular task is contingent on the route it takes and the plans of other agents. In this light making allocation decisions without taking into account routing and other agents' assignments may result in poor decision making. The final commonality is in time-oriented nature of the objective function in the domain. The speed with which agents can address tasks is the primary metric of quality; coordinated plans that result in long delays for agents are unlikely to be of high quality in such domains.

1.2 Coordinating agents in domains with intra-path constraints

In order to properly introduce our suite of approaches to coordination in domains with intra-path constraints we need to introduce some important concepts in multi-robot coordination. We then consider the space of possible ways to coordinate agents in our domains of interest, posing the problem of coordinating agents as a search problem. Finally, we discuss characteristics of domains with intra-path constraints that can potentially be exploited to enable approaches to tractably search for high quality coordinated behavior.

1.2.1 Coordination definitions

This section will describe some basic terminology associated with coordination a set of agents. We first consider what constitutes a plan for a single agent. Each potential agent plan can be viewed as an intended series of actions that can serve to transition the agent through a series of states. Whether or not a plan can actually be executed is a product of the agent's particular state and perceptions of the world, uncertainty in the environment, and potentially other agents' actions or states. An action sequence becomes a **schedule** when a particular time is associated with each action and state. In multi-agent domains we must consider not just individual agent schedules, but the schedules of multiple agents operating together in the same environment. Towards that end we next define the concept of a **coordination solution**. A coordination solution is a set of agent schedules, one for each agent operating in the domain. The coordination solution consists of the *intended* actions for each agent and the time at which the agent intends to perform those actions.

While many kinds of constraints can affect the ability of agents to execute the plans specified in a coordination solution, the factor of most interest to this work are inter-agent constraints. The presence of inter-agent constraints means that an agent's ability to transition to a desired state or to take a desired action is contingent on the actions that other agents perform or the states that they occupy. Some domains may not exhibit inter-agent constraints; we call these domains with **independent coordination**; the literature also refers to these domains as weakly coordinated, or involving only single robot tasks. In independently coordinated domains, once agent schedules are determined they should be able to execute those schedules without being affected by the actions taken by other agents. At the other extreme in terms of the presence of inter-agent constraints are domains where agent schedules are constantly mutually inter-constrained - these domains are referred to as requiring **tight coordination**, as defined by Kalra (Kalra, 2006). For an example of tight coordination requirements, consider a domain with an absolute requirement that agents maintain line-of-sight communications. In such a domain an agent's ability to take actions to move through the domain environment are contingent on the actions of other agents, who will also be attempting to move in the environment.

Domains with intra-path constraints fall somewhere between these two extremes, as inter-agent constraints exist but they are not constant. Instead, they occur only at particular times and places associated with constraint locations. In the disaster response domain, for instance, bulldozers must address intra-path constraints in order for fire trucks to achieve states that lie beyond constraint locations, and if bulldozers have not addressed those constraints by specified times fire trucks may be delayed in executing their schedules. While there is dependence at the schedule level, there is also substantial independence: the particular manner in which a constraint is addressed, including how the bulldozer reaches the constraint, or even which bulldozer addresses the constraint do not affect the fire trucks.

An important distinction for this dissertation is how inter-agent constraints are expressed in the agent schedules that comprise the coordination solution. If the schedules specify particular times and places where constraints will be satisfied and accurately account for inter-agent constraints, that solution is said to represent **explicitly scheduled coordination**. The alternative is **implicitly scheduled coordination**. An implicitly scheduled coordination solution does not reflect precise times or places at which coordination will occur; instead, agents will attempt to execute the next action in their schedule whenever the conditions to perform the action are met, and will otherwise wait. In the disaster response domain, in an implicitly scheduled solution fire trucks would select routes without considering the actions of bulldozers, and bulldozers would select constraints to address without considering the routes taken by fire trucks; fire trucks would simply wait until constraints along their routes were addressed before continuing to follow the routes. A central focus of this thesis will be to explore the benefits of using an explicitly scheduled approach versus an implicitly scheduled approach.

Determining the quality of a particular coordination solution is another important concept in this thesis. In order to evaluate coordination solutions we must consider the requirements of the **domain instance** the coordination solution attempts to address. A domain instance is an example of a kind of domain. In Section 1.1 we described a domain as involving a set of agents, tasks, and constraints; an instance of the disaster response domain would involve a particular set of agents, fires, and debris pile locations. The evaluation process for a coordination solution for a domain instance is to determine how well the coordination solution meets the requirements of the particular domain and instance for which it was developed. In our domains of interest, the primary **solution quality metrics** are time-oriented, and largely based on how quickly tasks are addressed and the importance and urgency of the tasks. One of the factors governing the performance of a coordination solution is how well agent intentions map to desirable behavior when executed. Evaluation thus involves some method to model schedule execution. Potential mechanisms for evaluation include simulating environments or coordinating real robots.

Another important concept in this work is the **coordination space** associated with a particular domain instance. The coordination space is defined as the set of potential coordination solutions for a given domain instance. The problem of determining a coordination solution can be viewed as a search within the coordination space for solutions that will achieve high quality. One primary approach used to limit the space that is actually considered during search is to employ a **planning horizon**. Using a planning horizon when searching the space means that the search only consider what agents will do for a particular period in the future; beyond the planning horizon agent actions are not considered during search. An important distinction in this work are **instantaneous assignment (IA)** versus **time-extended assignment (TE)** approaches, which differ in terms of planning horizon. An instantaneous allocation approach sets a single-task planning horizon for all agents, reasoning only about the next task each agent will address with no provision for future assignments. Time-extended assignment allows for robots to be assigned multiple tasks, extending the planning horizon future into the future.

The final concepts we introduce here concern the presence of uncertainty in the environment. In some cases it may be useful to treat a domain instance as fully known and unchanging; we call these domains **static domain instances**. In a static domain instance the coordination space associated with a particular domain instance will not change over time. If there is an expectation of uncertainty in the domain then instances are referred to as **dynamic domain instances**. In dynamic domain instances the coordination space will change over time as new information is learned about the nature of the domain instance.

1.2.2 Coordination spaces for domains with intra-path constraints

Having introduced a number of important concepts and definitions in the previous section, in this section we consider the coordination space for domains with intra-path constraints. In order to explore this topic, it is first useful to consider domains with independent coordination. In domains with only independent coordination requirements, the coordination space for a particular domain instance can be viewed as a two level hierarchy. At the top level of the hierarchy are potential allocations of domain tasks to agents, representing all potential divisions of domain responsibilities among agents. Given that each agent may be assigned several tasks in a given time-extended allocation, the second level of the hierarchy is the potential sequences of allocated tasks for each agent. For a given task allocation and sequence, the independence assumption states that only single agent planning is required to determine the best method for accomplishing a given sequence of tasks.

This two-level hierarchy is not sufficient to express the full space of potential coordination solutions in domains with intra-path constraints. To illustrate this point, consider the two agent example from the disaster response domain instance shown in Figure 1.4. Suppose a single fire has been assigned to the fire truck, and the goal of the truck is to reach the fire as quickly as possible. In an instance where there was no debris, the situation would be analogous to those found in independent task domains, where a single-agent planner could quickly determine the best plan to address the task. In this case the shortest distance path is for the fire truck to go down the

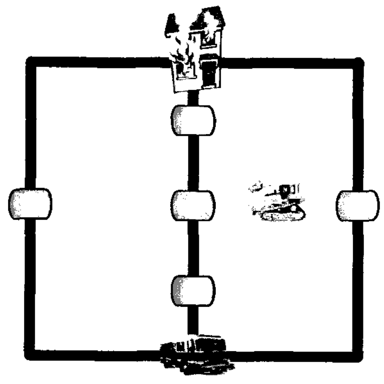


Figure 1.4: Example domain instance with a single fire truck attempting to reach a single fire. A single bulldozer is available to clear debris.

center road. When the intra-path constraints represented by the debris are considered, however, taking the path that is shortest in distance is unlikely to be the fastest plan, as there are a number of debris piles blocking that route. If we assume that all debris piles take the same amount of time to clear once reached by a bulldozer, whether or not the center path is the best route depends on the plan adopted by another agent - in this case the bulldozer. If we suppose the bulldozer has been assigned to clear all the debris piles in this instance, then its sequencing becomes the deciding factor in determining which route the fire truck should take. If the bulldozer elects to clear the right-most debris pile first, then taking the right route will allow the fire truck to reach the fire the most quickly; similarly, if the left debris pile is cleared first then the fire truck should take the left route. As a final point, if we consider the fire truck's goal in tandem with the bulldozer's sequencing decision, then the coordination solution that will allow the fire truck to reach the fire the most quickly is to have the bulldozer clear the right debris pile first, as the bulldozer is nearer to that pile and the left and the right routes are of equal length for the fire truck. Even if a task allocation and sequence are established for the fire truck and bulldozer the fire truck still must take into account the bulldozer's plan in order to determine the best route to the fire. Single-agent planning is not sufficient to determine the best plan given an assignment and sequence, and the routes that agents take become an important part of the coordination solution.

The space of coordination solutions for domains with intra-path constraints will have less uniformity than those associated with independent coordination domains, as the particulars of which part of agents' plans must be expressed in the coordination and the manner of the expression change from domain to domain, though there are substantial commonalities. In the next sections we detail the coordination space for our two primary motivating examples discussed in Section 1.1 within the context of a static domain instance.

The coordination space for disaster response

In this discussion we focus on the version of the disaster response domain with constraints of precedence. While there are a number of different ways to express the coordination space, we choose a method that most closely builds from the two level hierarchy for independent coordination domains. In disaster response with intra-path precedence constraints the coordination space still involves determine an allocation and sequencing of fire tasks to task-addressing fire truck agents. The set of potential task allocations and sequences only represent a portion of the potential space of solutions that must be considered. In addition, the route that each fire truck takes between tasks becomes a substantial aspect of the coordination solution, as well as the assignments and sequences of constraints for bulldozers.

There may be many potential routes a fire truck can take between tasks, though we can limit the space somewhat given two assumptions. First, we assume that we can represent the road network in a graph-based representation. Intersections, intra-path constraint locations, and task locations are represented as nodes in the graph. Edges in the graph connect these nodes, as it can be assumed that fire trucks can independently determine the fastest path between nodes using a single-agent planner. Edge costs can be represented by duration estimates produced by the single-agent planner. By this assumption, potential paths between two points can be determined using a graph-based planner in a relatively compact representation of the environment. Second, given the assumption that the goal is to get from one fire to another as quickly as possible we can assume that there is no benefit when moving from the location of one fire to another to repeatedly travel over a given segment. These two assumptions together yield a finite space of possible non-repeating paths from any location to any other location. For a single fire truck given an allocation and sequence, the space of possible plans needs to include each possible non-repeating route initially between the agent and the first fire in its sequence and then between successive fires in the sequence.

This space only reflects the fire truck plans however - the bulldozer plans must also factor into the coordination space. In this domain bulldozers' schedules are not constrained by intra-path constraints, and thus they can be assumed to be able to use single-agent planners to reach intra-path constraint locations. As their routes need not be represented in the space, the space of possible bulldozer plans can be represented in terms of potential constraint allocations and sequences.

While a two level hierarchy of allocations and sequences was sufficient to represent the coordination space in independent coordination domains, in the disaster response domain we need a five level hierarchy. We show potential decisions at each of these layers for an instance of the disaster response domain in Figures 1.5 through 1.8. In Table 1.1 we exhaustively enumerate the set of potential solutions at each level of the hierarchy hierarchy for both instantaneous allocation and time-extended allocation approaches. The top two layers are identical to those found in independent coordination domains - the potential allocations of tasks to task-addressing agents, and given allocations the potential task sequences. One potential task allocation is shown in Figure 1.5. Even

Hierarchy position	Instantaneous allocation	Time-extended allocation
1. Task allocation	$\frac{\lvert t\rvert!}{(\lvert t\rvert - \lvert a\rvert)!}$	$\sum_{i=0}^{2^{\lvert a\rvert}} \frac{1}{i!} \sum_{j=0}^{i-1} (-1)^j \binom{i}{j} (i-j)^{\lvert t\rvert}$
2. Task sequencing	1	$\prod_{i=1}^{\lvert a\rvert} \lvert t_i\rvert!$
3. Routing choices	$\prod_{i=1}^{\lvert a\rvert} Paths(t_{i0}, t_{i1})$	$\prod_{i=1}^{\lvert a\rvert} \prod_{j=0}^{\lvert t_i\rvert-1} Paths(t_{ij}, t_{i(j+1)})$
4. Constraint-satisfaction duty assignment	$\frac{\lvert c\rvert!}{(\lvert c\rvert - \lvert \beta\rvert)!}$	$\sum_{i=0}^{2^{\lvert \beta\rvert}} \frac{1}{i!} \sum_{j=0}^{i-1} (-1)^j \binom{i}{j} (i-j)^{\lvert c\rvert}$
5. Constraint-satisfaction duty sequencing	1	$\prod_{i=1}^{\lvert \beta\rvert} \lvert c_i\rvert!$

Table 1.1: Enumeration of the potential solutions at each level of the coordination space hierarchy for disaster response with intra-path precedence constraints. Lower levels of the hierarchy assume that a solution has been set at all upper levels of the hierarchy. t is the total set of tasks in the environment; a is the set of task-addressing agents; t_i is the set of tasks allocated to agent i; t_{ij} is the jth member in the sequence of tasks assigned to agent i; *Paths* is a function that produces the number of non-repeating paths between two locations; c is the set of constraints that lie along the selected routes; and β is the set of constraint-addressing agents.

for instantaneous allocation approaches, the space of potential single task allocations is large, and for time-extended allocations the space of allocations grows exponentially. This equation is for a Stirling number of the second kind, counting the ways to divide a set of t items into a subsets (Zlot, 2006). Given a task allocation, in the time-extended case the sequence of each agents' assigned tasks must be determined, with a factorial number of potential sequences for each agents' assigned tasks. One potential sequencing of the allocation shown in Figure 1.5 is shown in Figure 1.6.

The third level of the hierarchy are the potential non-repeating paths the trucks could take that allow the trucks to reach their assigned fires in the sequenced order; one set of potential routes is shown in Figure 1.7. In the instantaneous allocation case the potential routes to consider are the non-repeating routes between each agent's current location, represented by t_{i0}, and the location of its assigned task. In the time-extended case the routes between each task location in each fire truck's sequence must be considered. It is important to note that for each potential task allocation, sequencing, and routing we must consider task-addressing agents' actions in tandem, as

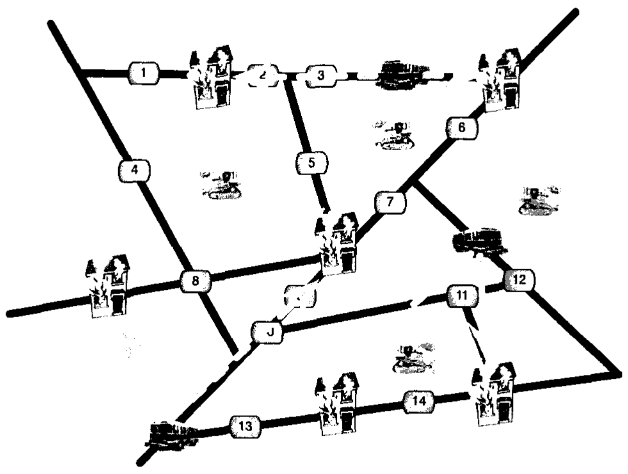


Figure 1.5: A sample allocation of fires to fire truck agents, representing one potential decision at the top level of the coordination space hierarchy for disaster response with intra-path constraints

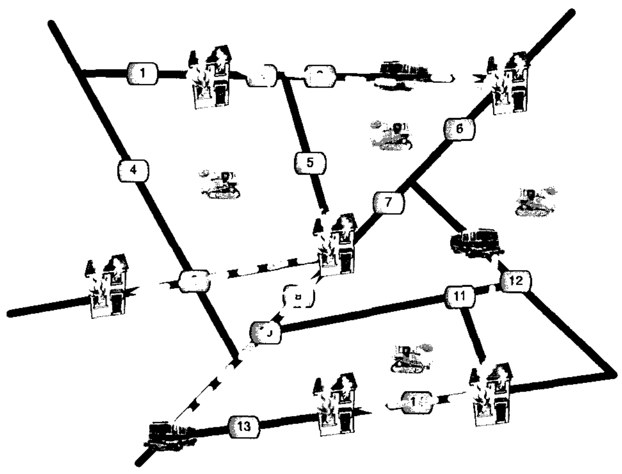


Figure 1.6: Given the allocation shown in Figure 1.5, one potential fire sequencing decision at the second level of the hierarchy

the decisions made for each task-addressing agent affect all other agents due to mutual dependence on constraint-addressing resources. For a given multi-agent path plan selected at the third level of the hierarchy, there is an allocation and sequencing of domain constraints to the constraint-addressing bulldozers that constitute the fourth and fifth layers of the coordination space hierarchy - an example given fire truck routes is shown in Figure 1.8. The set of constraints that must be allocated need only include those constraints that must be addressed given fire truck routes and not necessarily every constraint on every potential route.

The coordination space will not change substantially with the addition of other types of intra-path constraints, such as simultaneity requirements at either fires or debris piles. While more than one agent may need to be assigned to a task or intra-path constraint, the basic requirements of routing, task allocation, and task sequencing remain the same. The space of possible coordination solutions may not change with the incorporation of simultaneity constraints, but it may make searching the space a more difficult proposition.

The coordination space for on-order manufacturing

The coordination space for on-order manufacturing has some similarities and some differences from the space associated with disaster response. We first consider a version of the domain where mobile platforms have capacity for a single item at a time, where each staging area associated with an inventory tower can hold only a single item at a time, and where the only intra-path constraints are those of precedence, where a component must be in a staging area to be attached to an item.

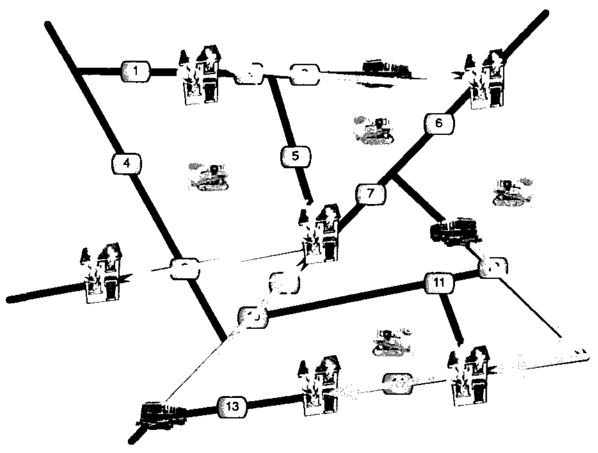


Figure 1.7: Given the fire sequences shown in Figure 1.6, potential routes that allow fire trucks to achieve those sequences at the third level of the coordination space hierarchy

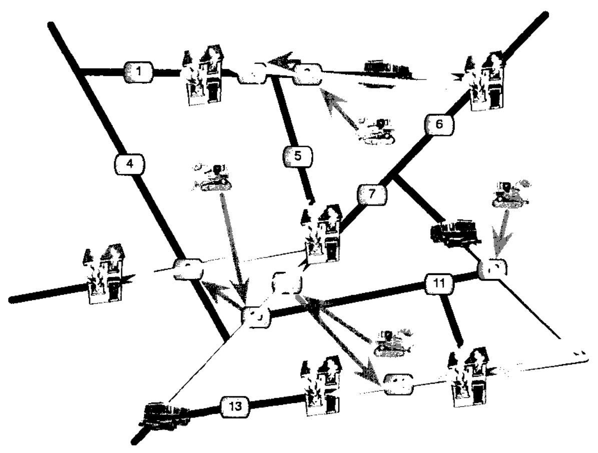


Figure 1.8: Given the fire truck routes shown in Figure 1.7, potential bulldozer constraint allocations and sequences at the fourth and fifth levels of the coordination space hierarchy

We first consider mobile platforms. At the top level there is still the problem of assigning and sequencing orders to mobile platform agents. Given a particular order, there are potentially a number of different sequential recipes for addressing orders. To fulfill the requirements of a recipe an agent must move to tower locations where the required parts can be attached, and then it must move to a shipping area to offload the item. The space of potential routes that will satisfy a recipe is governed by the potential locations at which each part can be acquired and then the set of shipping locations where a completed item can be offloaded. The coordination space also involves the plans of the tower agents. The tower agent must determine what parts to fetch and in what order.

The coordination space thus consists of a six level hierarchy. The top two levels consist of order assignments and sequences to mobile platform agents. Next come recipe selections given an order sequence. For a given recipe there is a space of possible routes which could be taken to acquire each part in order. Finally, there are the assignments and ordering of parts to be fetched by tower agents.

Variations on this domain can alter the coordination space considerably. If mobile platforms have the capacity to simultaneously construct multiple items, the sequencing decisions made by platform agents become more complicated than a simple queue. Instead both the order in which tasks are begun and the order which they are completed must be represented, subject to the capacity constraints of the agents. Increasing the capacity at staging areas or for tower agent payload capacity would have similar ramifications.

1.2.3 Implications for determining coordination solutions

The added levels of hierarchy required for representing the coordination space in domains with intra-path constraints make them substantially larger than those found in independent coordination, especially as some layers involve agent routes. This makes the challenge of tractably determining coordination solutions in such large coordination spaces significant. The size of the coordination space will almost certainly preclude using approaches that consider all possible allocations, routes, and sequences. Despite the challenges, there is substantial structure in the coordination spaces of domains with intra-path constraints that can potentially be exploited to achieve the goal of quickly determining high quality coordination solutions.

One form of potentially exploitable structure is spatial in nature. To illustrate, remember that we previously noted that given fire truck allocations, sequences, and routes, there still was the problem of allocating and sequencing constraints for bulldozers. Good solutions to the problem of allocation and sequencing constraints are likely to share similar structure, however. Fire trucks cannot follow their routes unless debris piles along those routes are cleared, and even if all debris piles are cleared later in the route a single debris pile blocking a fire truck's early in its route will prevent passage. As the goal is to allow fire trucks to reach fire locations as quickly as possible it is almost certainly more efficient to clear debris piles that occur early in fire truck routes before debris piles that will only effect the trucks later in routes. This observation can potentially be exploited in the form of a sequencing heuristic that could speed the search for constraint allocations and sequences that achieve high quality.

Another example of the potential uses of spatial heuristics can be seen in the coordination space for on-order manufacturing. While the space of potential routes for a particular mobile platform may include each different location at which a particular part can be procured, it may be possible to use spatial clustering algorithms that determine clusters of towers that are close to each other and can provide not just a single part but a series of parts. Identifying such a cluster can serve to focus search in the coordination space on routes that have both low travel time and thus potentially faster order completion. By reasoning about the spatial nature of the coordination space heuristics can be developed that ignore parts of the space that are unlikely to offer good solutions.

In our previous discussions of the coordination spaces for disaster response and on-order manufacturing we noted that the addition of simultaneity constraints does not substantially alter the coordination space. Adding simultaneity constraints can have profound implications for approach design, however; when schedules are more densely inter-related it becomes imperative to reason about the actions of all agents in concert. While considering agent schedules in concert can potentially lead to higher quality solutions, doing so involves searching in the full multi-agent coordination space; even for approaches with short planning horizons, search in the full multi-agent coordination space may be expensive in computational terms. Even if large numbers of multi-agent constraints inter-relate agent schedules there still can be independence between schedules that can be exploited;

by confining the search to areas that pertain to particular decisions by particular agents we can potentially exploit the substantial independence between agents to tractably search over longer planning horizons. While approaches that exploit independence may not discover solutions that improve overall system efficiency at the expense of the performance of particular agents, it may enable high quality solutions to be determined quickly.

1.3 Approaches to coordination for domains with intra-path constraints

In the previous section we described the coordination spaces associated with domains with intra-path constraints, and gave some potential insights into how spatial structure and independence can potentially be exploited to enable fast search that still determines high-quality solutions. In this section we consider the factors that can contribute to determining how to approach coordination in domains with intra-path constraints. Until now we have discussed solution quality metrics for evaluating the performance of a coordination solution, and considered domain descriptions and variation in terms of the type of constraints and the expectation of uncertainty. These are only a few of the aspects which must be considered; we use the concept of **scenarios** to encapsulate all the different factors that can affect approach design. For our purposes, the concept of scenario goes beyond the domain instance or specifications - it also involves extrinsic factors such as the time or computation available for determining coordination solutions, expectations of uncertainty, and availability of high-bandwidth networks. We next present a partial list of potential factors that may be involved in a scenario.

- Static domain instance characteristics
  - Type of intra-path constraints (precedence, simultaneity, agents involved)
  - Density of intra-path constraints
  - Ratio of task-addressing to constraint-addressing agents
  - Physical distribution of tasks, constraints, and agents
  - Reward function and distribution over tasks
  - Duration requirements of tasks and constraints
  - Size of the environment
  - Number of potential routes
- Expectations of uncertainty
  - Dynamic task issue
  - Durational uncertainty
  - Uncertainty in task/constraint locations or parameters
  - Perceptual uncertainty
  - Uncertainty in action

- Computation time availability
  - Initial wall-clock time available for computation
  - Time available for re-planning
- Resources available for computation
  - Number and speed of processors
  - Physical location of processors
- Communication resources
  - Available bandwidth
  - Reliability of communication
  - Cost of communication
- Distribution of information
  - Perceptual ambiguity
  - Time required for centralization of information
- Potential robustness issues
  - Robot death or incapacitation
  - Communication failure

Scenario factors can be roughly divided into two categories: domain characteristics and scenario resources and requirements. Domain characteristics represent the expected world state - the characteristics of the domain environment and the team of robots that will be deployed in the environment. Domain characteristics that can influence the approach selection are listed in the first two bullet points of the scenario factors list. The rest of the list are scenario resources and requirements; these are extrinsic factors to the domain characteristics that influence the form approaches must take and what resources are available for determining coordination solutions. In terms of time available for computation, if a scenario requires a solution to be determined quickly then a computationally expensive solution may be infeasible. The nature of the computational resources are another important factor in approach design. If a scenario has little time for computation in wall-clock terms, but many different powerful processors with fast connections between them, then using approach that employs **parallel computation** may be crucial. Parallel approaches are those where the search in the coordination space can be conducted in parallel on a number of different computers. The location of computational resources and the networks that connect them are other important factors in determining approach characteristics. Some scenarios may have limited bandwidth, or associate a cost with communication, making approaches that limit communication preferable. Communication also contributes to scenario requirements in terms of the distribution of information. In domains where agents have local perception capabilities and world state is imprecisely known, each agent will have its own view of the environment. A **distributed approach** to coordination determines plans for agents that each have their own view of the world without requiring **information centralization**, where the local perceptual states of agents are accumulated and reconciled into a central world view. Centralizing information can be costly in terms of time and communication.

In this dissertation we do not attempt to exhaustively account for all potential scenario factors in approach design. We instead focus on several factors. Evaluating relative approach performance in static domain instances is a central concern of this thesis; we focus particularly on maximizing the quality of the determined solutions while gauging the computational time required to determine solutions. We evaluate approaches in domains with different numbers of agents, densities and types of constraints, and sizes of environment in order to articulate how different approaches perform in domains with different characteristics. Another factor we consider is the effects of uncertainty on approach performance, and how approaches can be adapted to better cope with different forms of uncertainty. We seek to convey the strengths and weaknesses of approaches in terms of a number of scenario characteristics and requirements in order to both aid in approach selection and to offer insight into the challenges of coordinating agents in domains with intra-path constraints.

Our suite of approaches, summarized in Chapter 3.1, vary in terms of the implicitness or explicitness of the scheduling in produced coordination solutions; the planning horizon employed during search; the reactivity to new information; and the techniques used to search the space of possible coordination solutions. We present five different approaches during the course of this dissertation. Four of the approaches use market-based methods to determine coordination solutions. Market-based methods are an intuitive and useful way to structure heuristic and greedy search in the space of potential coordination solutions. The market-based approaches heavily exploit domain features in order to determine good solutions relatively quickly. Our last approach uses a very different methodology, randomized evolutionary search, to search in the full coordination space of a domain instance; the requirements in terms of computational resources and available time are high for this approach, but it can potentially determine higher quality solutions than the market-based approaches.

1.4 Thesis statement

Determining coordination solutions in domains with intra-path constraints represents a significant and important challenge that is not well addressed by current multi-robot coordination techniques. This thesis asserts that the structure of these domains can be exploited to efficiently produce high quality coordination solutions. We present a range of approaches to coordination; in each approach we capitalize on domain structure in order to enhance solution quality while minimizing computational requirements. Selecting an approach that is best suited for a particular scenario requires considering a number of different factors, including the characteristics of domain instances, expectations of uncertainty, available resources for planning, and scenario requirements.

1.5 Thesis road map

The next chapter presents literature related to multi-robot coordination domains with intra-path constraints. We then summarize our approaches and introduce factors in approach selection and evaluation in the first part of Chapter 3. The remainder of Chapter 3 describes our two least computationally expensive approaches to coordination. Chapter 4 introduces our more computationally expensive market-based approaches, which search more comprehensively in space of possible coordination solutions and employ longer planning horizons. Chapter 5 describes our randomized evolutionary approach to search in the full coordination space, and also contains our most comprehensive comparison of all five approaches in static domain instances with different parameterizations. In Chapter 6 our discussion expands to dynamic domain instances; this chapter describes how our approaches can be adapted to cope with significant sources of domain uncertainty and evaluates how uncertainty affects the performance of the approaches. Chapter 7 examines domains with intra-path constraints that may not be exclusively constraints of precedence; we also discuss how our work can be adapted to the somewhat different requirements of collaborative resource acquisition domains. Finally, Chapter 8 summarizes the findings and contributions of this thesis and discusses potential future work.

Chapter 2 Related work

In our introductory chapter we detailed the coordination spaces of domains with intra-path constraints, as well as detailing scenario factors that must be accounted for in approach design and deployment. In this chapter we look to the literature to examine what potential approaches already exist and to evaluate their abilities to address scenario requirements for domains with intra-path constraints.

An important distinction between approaches we introduced in Chapter 1.2.1 was between explicitly and implicitly scheduled coordination approaches. An approach that produces implicitly scheduled coordination solutions makes decisions about agent plans without considering and explicitly scheduling the times and places that interactions will occur. Making allocation, routing, and sequencing decisions without fully considering inter-agent constraints may lead to relatively poor performance in some circumstances, but may be appropriate for domains where little time is available for coordination or where expected uncertainty levels are high. The literature provides many approaches that may guide implicitly scheduled coordination. Fewer approaches capable of determining high-quality explicitly scheduled coordination solutions exist. Specifically, we are interested in approaches that can quickly determine high-quality, explicitly scheduled coordination solutions. Determining such solutions requires searching in the space of potential assignments, sequences, and routes for task- and constraint-addressing while attempting to use a minimum of computation. A final class of approaches of interest are those that may use a good deal of computation, but can potentially search more comprehensively in the coordination space and determine solutions with higher quality.

Before examining the literature, we need to position our categorization of related work in the context of the widely adopted nomenclature for classifying multi-robot coordination approaches: the three-axis nomenclature introduced by Gerkey and Matarić (Gerkey & Matarić, 2004). Their first axis is single-task robots (ST) versus multi-task robots (MT): ST denotes single-task robots, which are capable of addressing only a single task at a time, while multi-task robots may be able to address a number of tasks concurrently. The second axis is single-robot tasks (SR) versus multi-robot tasks (MR): single-robot tasks are addressable by a single robot, while multi-robot tasks can require more than one robot. The third classification axis is instantaneous versus time-extended assignment, a distinction we covered in Chapter 1.2.1.

The Gerkey and Matarić nomenclature is very useful, and we make particular use of the distinction between instantaneous versus time-extended assignment. However, by their own accounting this nomenclature does not factor in intra-task constraints; furthermore, only the umbrella category of multi-robot tasks exists for describing inter-agent constraints. Certainly tasks with simultaneity requirements fall under their definition of multi-robot tasks, but the relationship between task-addressing and constraint-addressing agents in our domains of interest is not well captured in this nomenclature. In the disaster response domain with intra-path precedence constraints, the efforts of a single robot are sufficient to address both tasks and constraints, so the multi-robot task designation does not seem to apply. At the same time, inter-agent constraints affect agents' execution of their schedules, making tasks require the efforts of multiple robots. For our purposes the most relevant categorizations of related work concern the treatment of inter-agent constraints, and the Gerkey and Matarić nomenclature is insufficient to capture the differences between approaches that reason about different kinds of inter-agent constraints.

We divide our discussion of related work into four sections. The first section considers work in independent coordination domains; this work does not address inter-agent constraints and can only inform the generation of implicitly coordinated approaches. The second section considers domains where constraints may exist between tasks, but no inter-agent constraints exist. The third section discusses work that specifically focuses on inter-agent constraints. The final related work section considers relevant work in tightly coordinated domains.

2.1 Independent coordination approaches

The first substantial body of work we consider is the literature associated with domains with independent coordination requirements. We divide this work into two segments - approaches that do instantaneous assignment and those that do time-extended assignment. We focus on market-based approaches to independent coordination, as these are a primary technique we will employ in our own approaches.

2.1.1 Instantaneous allocation approaches

As noted by Gerkey and Matarić, instantaneous allocation in independent coordination domains is essentially trying to solve an optimal assignment problem (Gerkey & Matarić, 2004). Vail and Veloso use a market-based approach to do role-assignment in the RoboCup soccer domain with homogeneous agents (Vail & Veloso, 2003). Gerkey and Matarć (Gerkey & Matarić, 2002) also use a market-based approach for instantaneous assignment of hierarchical task structures. Simmons et al. (Simmons *et al.*, 2000a) use a market-based approach for instantaneous assignment in a mapping and exploration context.

2.1.2 Time-extended allocation approaches

The TraderBots approach uses a market-based approach with sequencing for a wide variety of ST-SR-TE domains and objective functions (Dias, 2004) (Dias *et al.*, 2004b) (Zlot *et al.*, 2002). TraderBots has additionally been shown to be robust under a variety of failure conditions, including communication failures and partial and full robot malfunction (Dias *et al.*, 2004a). Berhault et al. use a combinatorial auction approach for time-extended assignment of exploration tasks with a travel cost metric (Berhault *et al.*, 2003). Lagoudokis et al. prove theoretical bounds for using sequential single-item auctions for time-extended assignment of independent single robot routing tasks under a variety of different metrics (Lagoudakis *et al.*, 2005) (Lagoudakis *et al.*, 2004).

Independent coordination domains can potentially include constraints on tasks that are not inter-agent constraints, such as task deadlines or time windows. In our previous work we looked at single robot tasks with time-based rewards and deadline constraints, with penalties assessed for failed commitments (Jones *et al.*, 2007). We focused particularly on oversubscribed domains, where robots could not accomplish all tasks even if acting optimally. We chose to use a market-based approach similar to TraderBots, but noted that in domains where agents must commit to tasks at the time of issue and where tasks vary in importance and urgency that a conventional market-based approach resulted in agents committing many penalties. We then supplemented the conventional market-based approach with a learning mechanism that allows agents to use previous experience to modify their bids for tasks to better account for the fact that other more important and urgent tasks may be issued in the future. Melvin et al. consider ST-SR domains where tasks have associated time-windows for execution (Melvin *et al.*, 2007). In these domains robots receive a reward if they occupy the location of a task during the time window. Melvin et al. provide an optimal approach for two special cases of the domain, and offers a heuristic approach for general cases of the domain.

Enright et al. consider a vehicle routing domain where the goal is to reach dynamically appearing target points so as to minimize the expected delay in reaching a target task after it appears (Enright *et al.*, 2009). Making the problem more complicated is an assumption that vehicles must obey parameters for motion corresponding to a Dubins vehicle model, a standard motion model for unmanned aerial vehicles. They examine low and high load cases, which differ in the expected rate of dynamic task issue. They develop policies for coordination and prove bounds in both high and low load cases. Smith et al. examine a similar domain, with an added complication that tasks fall into different priority classes (Smith *et al.*, 2009). Different priority classes may require different times to service upon arrival, and the goal of the domain is to minimize service times given the relative importance of classes.

One interesting segment of work that deserves separate treatment is Zlot's work on task allocation for domains with complex tasks (Zlot, 2006). Zlot's definition of complex tasks are tasks that cannot be directly executed by any robot in a domain, but must instead be decomposed into tasks that robots can independently execute. A decomposition of a complex task is a breakdown of the task into a task tree; the task tree has the complex task as a root node, and a number of children nodes that, if all satisfied, will satisfy the complex task. Child tasks can either be inter-related with AND connectives, which specify that all children must be satisfied to satisfy the parent nodes, or OR connectives, which require that only a single child be satisfied to satisfy the parent nodes. Child tasks can be complex, meaning they in turn must be decomposed, or simple tasks, which Zlot assumes must be within the capabilities of a single robot. Zlot also makes an important assumption that for every task there is some robot in the domain that can decompose the task completely into tasks the decomposing robot could independently execute. If this assumption is not met the complex task would go unassigned. After the decomposition phase, his method supports assigning decomposed nodes through re-auctions, and other agents can also selectively re-decompose parts of the tree, purchasing complex task nodes if their re-decompositions produce lower-cost solutions.

Zlot's approach, while highly relevant, is not sufficient to determine explicitly scheduled coordination solutions in domains with intra-path constraints. The ultimate output of a decomposition is a task tree with AND/OR connectives, with single agent tasks at leaves. Once the task tree is determined, the problem is one of independent coordination as agents' schedules are not inter-related. Furthermore, the decomposition methodology proposed in this work depends on the ability of single agents to fully decompose and schedule the sub-tasks associated with a complex task. During the auctioning process, each complex task can only be sold to an agent that can both fully decompose the task and then can produce an independent fitness estimate for each simple leaf node. In our domains of interest, agents cannot determine fitness for tasks in isolation, as their schedules are highly dependent on the actions of other agents involved in addressing other constraints or tasks. Due to the inter-related nature of schedules, agents cannot accurately assess the their fitness for a task without consulting other agents. Zlot's decomposition framework will not support the interconnections required to obtain high efficiency solutions in our domains of interest.

While Zlot's approach does not fully address our domains of interest, he makes an important observation that we carry forward into our work. Zlot recognizes that work in task allocation for complex tasks generally conforms to one of two approaches - some work decomposes tasks first, and then tries to allocate the tasks based on the decompositions, while other work assigns tasks to agents that then need to decompose them as best they can. Both of these approaches fail to recognize that the best possible solutions will optimize the decompositions based on assignment as well as optimizing allocations based on decompositions. Zlot's approach simultaneously considers both decomposition and allocation, improving the efficiency of the search and the quality of the determined coordinated behavior. In our problem domains, assignment of complex tasks must depend on the scheduling of agents, which depends on the allocation of interactions in sequential plans - all must be considered simultaneously to determine the objective-function maximizing solution.

2.1.3 Implications for our domains of interest

The literature addressing independently coordinated domains does not support the consideration of inter-agent constraints during search in the coordination space, and will not help us determine high-quality explicitly scheduled coordination solutions. However, we can potentially use some techniques used in the this literature to develop implicitly scheduled approaches to coordination. Market-based approaches to independent coordination, whether using instantaneous or time-extended assignment, can generally determine reasonable solutions quickly and in a partially distributed fashion. There is the potential to apply these approaches to quickly determine reasonable implicitly scheduled approaches while using a bare minimum in terms of computation time.

2.2 Approaches for domains with inter-task constraints

The next segment of literature we will examine are domains where there are constraints between tasks, such as ordering constraints that specify that tasks need to happen in a certain order, or time-oriented constraints, such that tasks collectively need to be addressed in a certain interval.

2.2.1 Multi-robot coordination literature

Mackenzie considers tasks with temporal and ordering constraints between tasks - for instance, a certain group of tasks may need to be closely temporally spaced and occur in a particular order (MacKenzie, 2003). He uses a market-based approach to coordinate agents. In this approach tasks are temporally constrained, and agents incorporate the constraints into their bids in the form of multi-dimensional cost functions. By bidding cost functions instead of single fitness values it equips a central auctioneer agent to determine not only which agents should be allocated tasks but other coordination decisions, such as when particular tasks will be performed. Lemaire et al. also focus on task-level temporal constraints for single-robot tasks (Lemaire *et al.*, 2004). Their approach is market-based, and can cope only with constraints phrased in terms of a certain task occurring some time period before another task. Their approach permits the handling of precedence constraints, but ignores simultaneity constraints. Botelho et al. also focus on tasks with precedence constraints in the design of their system M+ (Botelho & Alami, 1999). Their chosen method is to simply signal when tasks have been begun and ended, so that robots assigned tasks later in the precedence order are aware that task execution can begin. Shima et al. use a genetic algorithm approach to allocate and sequence a set of tasks with precedence and timing requirements to a team of unmanned aerial vehicles (Shima *et al.*, 2006). They show their approach outperforms a randomized approach and can outperform a breadth-first-search approach for given algorithm run times on small problems.

2.2.2 Vehicle routing literature

Another segment of the literature that concerns itself with tasks with inter-task constraints can be found in the large body of work addressing Vehicle Routing Problems, or VRP. The basic formulation of a VRP consists of a number of vehicles picking up and distributing goods to a number of delivery locations (Toth & Vigo, 2001). There are a huge number of VRP application variants, many of which place one or more constraints on the basic VRP - for instance, capacity constraints on agents, or time-windows on delivery; other variants include prioritizing among deliveries, or assessing rewards or penalties based on performance. The goal is generally to optimize in terms of costs, such as travel distance, or to maximize rewards while obeying system constraints; other goals may involve providing a certain level of service with a minimized number of delivery vehicles, or minimizing penalties for poor service. The problem can either be formulated as a static problem, where all deliveries are known, or an online problem where new tasks arrive periodically. Most VRP literature involves precedence constraints of some kind, as shipments, packages, or passengers must be picked up before they can be delivered.

VRP literature is relevant to our work, as it involves task allocation, routing, and scheduling. For an an overview of variations of the general VRP see Toth & Vigo (2001). We will examine one application variant - the Dial-A-Ride domain - more closely to illustrate the relevance and limitations of VRP approaches to our domains of interest. The Dial-A-Ride problem consists of routing and scheduling for an on-demand transportation system - a number of users in a city request pickup from particular locations and desire to be transported to particular destinations (Cordeau & Laporte, 2007). The goal is to accommodate a very large number of passenger requests, and to determine allocations of passengers and routes for a set number of transportation vehicles that maximize quality of service. The problem can either be static, with a known set of passengers, or dynamic, where passenger requests arrive periodically. Dial-a-ride variations can include different measures of quality of service, time windows, capacity constraints on transportation vehicles, and constraints on the maximum time any single passenger is on the bus. Exact approaches based on branch and cut algorithms exist for variants of the static Dial-A-Ride problem that can accommodate up to 100 passengers (Ropke *et al.*, 2007), but due to tractability concerns most approaches even for the static problem utilize different heuristics and may not guarantee optimality. Of special interest are approaches that use local search methods to generate high quality solutions in large problem domains relatively quickly. Jørgensen et al. use a genetic algorithms approach to assigning passengers to vehicles, and then a heuristic approach to routing based on the assignments (Jørgensen *et al.*, 2007). Melachrinoudis et al. use a tabu search method to transfer passengers between routes after an initial heuristic assignment period (Melachrinoudis *et al.*, 2007).

2.2.3 Implications for our domains of interest

Approaches to domains with inter-task constraints represent a step towards representing the complexity of agent interactions in domains with intra-path constraints, but these approaches are not equipped to efficiently determine high-quality explicitly scheduled coordination solutions in our domains of interest. While constraints can inter-relate tasks, for instance requiring that certain tasks be performed before others, none of these approaches support reasoning about making routing decisions based on the constraint-satisfying actions of other agents. In the vehicle routing literature, for instance, while constraint requirements must factor into the sequencing decisions made in addressing a series of tasks, agents need not interact while executing their path plans. While these domains require reasoning about precedence constraints, the type of reasoning that is done still would only yield implicitly scheduled solutions in our domains of interest.

2.3 Approaches for domains with inter-agent constraints

In this section we address the most directly applicable area of related work: work that addresses coordination in domains with inter-agent constraints. We first consider approaches that reason about relatively short time horizons. We then focus on approaches that extend the planning horizon, but may not guarantee optimal solutions or search in the full space of possible coordination solutions. Finally, we consider approaches that could potentially offer optimal performance, which requires searching in the full coordination space.

2.3.1 Short time horizon approaches

We divide short time approaches into two application categories - related work in the RoboCup Rescue Simulation League, and the literature in single-task or partially time-extended coalition formation.

RoboCup rescue simulation league

The RoboCup Rescue Simulation League is a competition with a goal of coordinating heterogeneous agents for disaster response in the wake of a large city-wide disaster (Kitano *et al.*, 1999).

In the RoboCup Simulation League there are 6 types of agents:

- FireBrigades, which can extinguish fires
- PoliceForces, which can clear roads
- AmbulanceTeams, which can rescue civilians
- Center agents for each of the above three types, which coordinate all the agents

FireBrigades and AmbulanceTeams cannot use roads until they have been cleared by PoliceForce agents, a feature similar to that in our own disaster response domain. The competition involves around 100 total agents, but the number of agents is intended to rise as computer hardware improves. There are many factors that contribute to the complexity of the competition, including resource limitations, real-time constraints on deliberation, limited perception, limited communication, agent interference, and unknown or changing task characteristics (Paquet, 2006).

The competition has spurred research not only into task allocation, but also in human-agent and human-robot interaction, and adjustable autonomy (Schurr *et al.*, 2005). The work of greatest relevance are those that specifically addresses task allocation. Nair et al. implement both a centralized combinatorial auction approach which uses time-extended allocation and a less communication-intensive distributed instantaneous allocation approach where agents make decisions exclusively based on local reasoning (Nair *et al.*, 2002). In the combinatorial auction approach, agents bid exclusively based on travel to tasks without reasoning about whether better routes might be found if PoliceForce agents cleared particular roads. Suarez et al. also use a combinatorial auction approach where bids are conditioned only on travel distance (Suàrez *et al.*, 2005). They include a mechanism for re-scheduling tasks when obstructions are discovered during execution; the mechanism involves either determining an unobstructed route or informing the PoliceForces center agent that it would like a road cleared. While the combinatorial auction approaches of Nair et al. and Suarez et al. are both time-extended in that multiple tasks can be assigned to each agent, in neither do agents fully consider what interactions they may require during allocation.

Paquet presents a number of his RoboCup Rescue Simulation League approaches and results in his doctoral dissertation (Paquet, 2006). First, he uses an online **partially observable Markov decision process (POMDP)** algorithm to select actions for PoliceForces agents. An important part of the approach is his design of a local reward function, which gives high reward for clearing roads where blockages affect many agents, and achieves dispersion by punishing agents for being near each other. The PoliceForces agents each run the online POMDP in parallel, updating local beliefs as to the position of other agents and other world data. By using only factored POMDPs, pruning off significant portions of the belief space during search, and leveraging a number of simplifying assumptions the algorithm is made efficient to run within the real-time constraints imposed by the RoboCup competition. This POMDP approach is well-suited to the RoboCup domain in that it explicitly models uncertainty, and implicitly performs task allocation, as agents independently make decisions about which tasks to address. We do not believe that this approach will extend to our domains of interest, however. The precedence interactions in our domains of interest mean that agents' decisions are highly interdependent, and individual agent POMDPs will not support this level of interaction.

Coalition formation

We next consider algorithms that take instantaneous or partially time-extended approaches to allocation for multi-agent tasks. This approach style generally falls into the ST-MR-IA segment in the Gerkey and Matarić nomenclature. The problem of determining what groups to form to address tasks is frequently referred to as coalition formation, as groups or sub-teams of agents must combine efforts to address tasks that require capabilities beyond those possessed by any single robot.

Vig and Adams (Vig & Adams, 2006) and Tang and Parker (Tang & Parker, 2007) all use coalition formation algorithms with instantaneous assignment. In both works, coalitions are created by forming disjoint sub-teams and matching those sub-teams to eligible tasks so as to maximize an objective function. Any tasks that go unassigned during this process are deferred until some agents finish with their previously assigned tasks. When some agents are ready for a new task, an assignment period can occur with the tasks that remained from the previous assignment as well as any newly issued tasks. While both approaches share some basic features, there are substantial differences. Vig and Adams focus on producing a Balance Coefficient to better distribute tasks among different coalitions, while Tang and Parker use a schema-based approach that emphasizes not only the forming of coalitions but also how the coalition members should coordinate to accomplish their assigned tasks. Tang and Parker's approach additionally includes a mechanism for task assignment only after all relevant preconditions have been satisfied, equipping it to allocate tasks with preconditions. In follow-on work Tang and Saha improve single-task allocation given a large number of potential coalitions using an anytime algorithm (Tang & Saha, 2008).

Another approach related to coalition formation is our work in dynamic sub-team formation for pickup teams (Jones *et al.*, 2006). In this work we combine market-based role allocation with a tight coordination mechanism based on a notion of formulating plays to address tasks. Plays are essentially pre-programmed recipes for performing particular tasks that may require that heterogeneous robots tightly-coordinate to perform a particular action (Bowling *et al.*, 2004). Each play specifies a set of roles, with each role requiring particular capabilities. The play specifies how the robots filling the roles interact to accomplish the requirements of a task. Many plays can match a particular task. In our work a market-based allocation system is used both to allocate robots to roles and also to determine which of many matching plays should actually be executed.

Lin and Zheng use a combinatorial auction to instantaneously allocate non-decomposable multi-robot tasks to a group of robots (Lin & Zheng, 2005). They centralize capability information at the auctioneer, which tentatively awards the tasks based on capabilities and cost functions. The winning bidders must then determine a group plan for the task; once they have accomplished this the award is confirmed. Sariel and Balch use instantaneous assignment for coalitions for resource- and precedence-constrained tasks, focusing on dynamic reassignment when task conditions change or new tasks are discovered (Sariel & Balch, 2006). Sariel et al. build on their previous work by using a method for generating rough schedules for tasks in the form of priority queues with priority determined by approximate cost estimates and precedence relationships (Sariel *et al.*, 2007). Their approach does not actually schedule tasks or form coalitions for any tasks other than those that our under consideration for immediate execution, and is thus a hybrid approach between instantaneous and time-extended assignment. Guerrero and Oliver use a learning-based system to account for inter-agent interference in determining coalitions and assignments in a cooperative transport task (Guerrero & Oliver, 2006).

Another hybrid approach between instantaneous and time-extended assignment is the token-based approach Scerri et al. use for task allocation (Scerri *et al.*, 2005). In this work they explore an approach that permits both the allocation of simple tasks which require only a single agent with a particular level of capability and access to resources, and AND tasks, which require simultaneous execution by a number of agents. Their approach is to generate tokens that represent both task assignment and access to necessary resources; they extend this approach to allocating AND tasks by the use of potential tokens, which represent a commitment to participate in the execution of an AND task when all necessary agents become available. Acceptance of a potential token may not require immediate action, and agents can perform other tasks in the meantime. The use of potential tokens makes this a hybrid approach between instantaneous and time-extended assignment; agents may commit to perform a number of tasks, making the approach more time-extended than purely instantaneous assignment approaches; at the same time, agents do not explicitly schedule AND tasks upon commitment, making this approach not fully time-extended. The token-based approach is shown to perform well in a variety of domains, especially in terms of communication costs, achieving slightly reduced performance versus a greedy centralized approach while using several orders of magnitude fewer messages. The reduced communication usage allows this approach to be plausibly used in domains with thousands of agents even if communication bandwidth is limited.

We argue that these approaches are insufficient to tractably determine high quality explicitly scheduled coordination solutions in domains with intra-path constraints. For one thing, to determine the value of a coalition – say a set of fire trucks and bulldozers in the disaster response domain – for a particular assignment requires solving a complicated routing and constraint sequencing problem. Determining coalition value is expensive in our domains of interest, and approaches that depend on being able to easily determine this value will not be easily applicable. Furthermore, considering exclusively single task assignments may not be sufficient for determining explicitly scheduled solutions if a large number of constraints separate agents from tasks. While reasoning about time-extended task allocation is not necessary for determining explicitly scheduled coordination solutions, time-extended constraint allocation is required. Finally, these methods do not consider the problem of searching the space of potential routes for task-addressing agents given constraint requirements. While short time horizon, coalition formation algorithms can potentially offer insights in coping with simultaneity requirements, these approaches do not fully address the problem of tractably determining explicitly scheduled coordination.

2.3.2 Longer time horizon approaches

We next consider longer time horizon approaches that explicitly address multi-agent tasks. We consider tasks by application area, first discussing the literature related to hierarchical task networks and then discussing the literature related to multi-agent routing.

Hierarchical task networks

One substantial body of work associated with time-extended assignment involves representing complex tasks as **hierarchical task networks (HTNs)** (Erol *et al.*, 1994), which depict tasks in terms of decompositions and inter-relations between tasks. HTNs consist of decompositions of complex tasks into other complex tasks or simple tasks that can be individually executed by an agent. A fully decomposed HTN will have all simple leaf nodes. HTNs can include connectives that relate branches of the network: AND connectives indicate that all branches must be satisfied to satisfy the parent, while OR connectives mean that any of the branches can be satisfied to satisfy the parent. Other connectives such as XOR can be used as well. Tasks in the HTN can also be associated with constraints - constraints can include time windows for execution, precedence relationships with other tasks, or other pre- or post-conditions based on resource constraints. Zlot's task trees, described in Section 2.1.2, are HTNs with branch connectives but no associated constraints.

Planning for HTNs can include determining the tree decomposition for complex tasks and also the allocation and scheduling of simple leaf tasks such that network requirements are met and high quality is achieved. While HTN planning approaches are potentially useful in our domains of interest, one obstacle to using existing approaches to HTN planning is that most of the approaches in the literature focus on fully decomposing HTNs, or allocating and scheduling the decomposed HTN, but rarely address both of these stages. The DARPA COORDINATORS project, for instance, looks at allocation of an existing HTN to a team of agents where none of the agents can see the entire HTN at any given time (Sims *et al.*, 2006) (Zimmerman *et al.*, 2007) (Smith *et al.*, 2007). This work focuses exclusively on allocation and scheduling a fully decomposed HTN. Similarly, substantial effort has been put into scheduling HTN networks by modeling problems as distributed constraint optimization problems (DCOPs) (Sultanik *et al.*, 2007). Once a problem has been formulated as a DCOP then general DCOP solvers can be employed. This work again focuses on scheduling a pre-existing task network.

Another part of the HTN literature focuses on a particular form of decomposed HTN: supply chains. In supply chain problems customers place orders for products – a computer, for instance – to be delivered on a specified date for a price specified by the seller. The supply chain indicates how to assemble the computer, which may require acquiring different kinds of parts from different locations, performing all actions necessary to assemble the computers, and shipping to a retailer's location. To assemble the computer requires negotiating complicated task interactions - precedence-related tasks must be scheduled, with a goal to optimize for speed and profitability. The form of the supply chain is fully determined, making this an HTN planning problem with self-interested agents but no decomposition. The MAGNET system (Collins *et al.*, 2002) is a combinatorial auction based approach to supply chain allocation, focusing on bid selection (Collins *et al.*, 1999), and scheduling based on proffered bids (Babanov & Gini, 2006). Other work on supply chains focuses on determining the order in which negotiations occur in the network (Zhang, 2002) (Zhang *et al.*, 2005). By determining better ways in which to order task negotiations the authors show that they can improve the quality of the planning solution.

Only one reference in the HTN literature focuses on how hierarchical task network formation can impact allocation and scheduling: the work of Hunsberger and Grosz evaluates how the allocation and scheduling may impact the choice of network decomposition (Hunsberger & Grosz, 2000). In their work they suppose that they have been given a series of HTN "recipes" for addressing a complex task, and need to determine which recipe will best address the task given agents' existing commitments. In order to determine the quality of the recipe they must allocate and schedule all simple tasks specified by the recipe; the approach they take to allocation is to use a combinatorial auction. To alleviate exponential computational cost of performing the combinatorial auction they first divide the actions in the recipe into roles, where any robot fulfilling the role will be obligated to complete all the tasks in the role. Clustering tasks into roles can save substantially in terms of computation, as it heuristically reduces the search space. Agents then form bids for roles based on their existing commitments, and can submit bids with time conditions as to when the actions in the role will be performed. A modified winner determination algorithm for combinatorial auctions is used to award tasks based on the bids with their associated time constraints. This procedure must be followed for each possible recipe for the task.

Generating efficient solutions in our domain is in some ways more difficult and in some ways easier than for the fully determined task networks described above. Our domains are more difficult in that we are not given a pre-existing network, and must both determine the form of the network and then how best to allocate and schedule the selected network. Our domains are easier in that the associated networks are not of arbitrary complexity; instead, the networks have distinct form and a good deal of exploitable structure. We first consider in more depth the challenge of determining coordination solutions in our domains of interest using existing HTN approaches. To represent domains with intra-path constraints in an HTN it is important to account for the potential routes that task-addressing agents take between their assigned tasks; an agent's dependency on particular constraints being satisfied is a product of its selected route, and in order to formulate valid task networks for allocation and scheduling this selective dependency must be represented. Even given task allocation and sequences, the potential decompositions of tasks into task networks requires representing the full space of routes between tasks as well as the resulting dependencies for constraints along those routes. This space of potential HTNs will almost certainly be too large to search exhaustively.

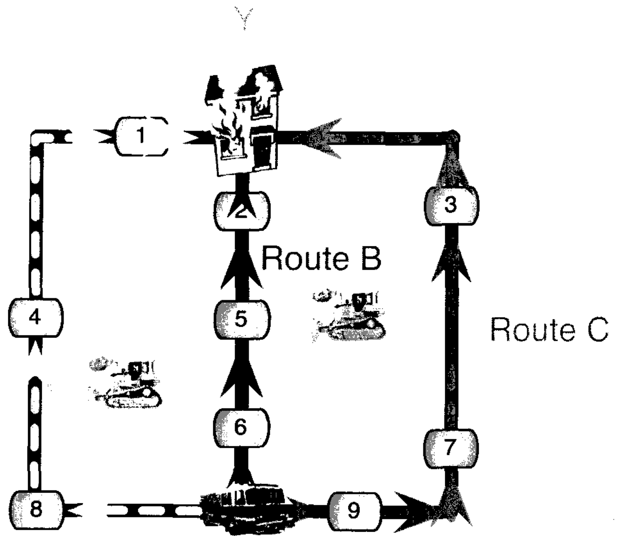


Figure 2.1: An example instance in the disaster response with intra-path constraints domain illustrating the three potential routes that a fire truck can take to reach Fire Y.

To illustrate the problem of representing the space of potential plans as HTNs consider the example shown in Figure 2.1. In this example there is only a single task, a single fire truck, and three potential non-repeating routes the fire truck could take to reach the fire. If we take a HTN view of the problem of addressing the fire, the problem can be decomposed into the network shown in Figure 2.2. Each route represents one method by which the fire Y can be addressed; an OR relationship exists between the potential routes, illustrated by the large dotted lines in the graph. In order to follow one of the routes, further decomposition is required into subcomponents - the fire truck must drive the route, and constraints along the route must be satisfied. Each leaf node must be satisfied to satisfy the parent task corresponding to a particular route. The precedence relationship between the constraint-addressing duties and the fire truck routing task is represented by the small dotted black arrows - these indicate that the completion time of the routing task depends on the completion times of the constraint satisfaction tasks. Given this decomposition, any general HTN solver could determine the best method to satisfy the root node, selecting a route node and allocating and scheduling tasks in the decomposition so as to satisfy the requirements of the tree and obtain high reward.

As we show in this example, each different route for a single fire truck agent to reach and address a single fire represents a distinct subtree within the HTN, and the set of constraint satisfaction duties that must be allocated is different for each route. For larger domain environments the space of potential routes can be much larger than the 3 non-repeating paths shown in Figure 2.1. The problem is further complicated if there are many different potential task assignments for each fire truck operating in the domain. Even if the potential allocations are restricted to instantaneous

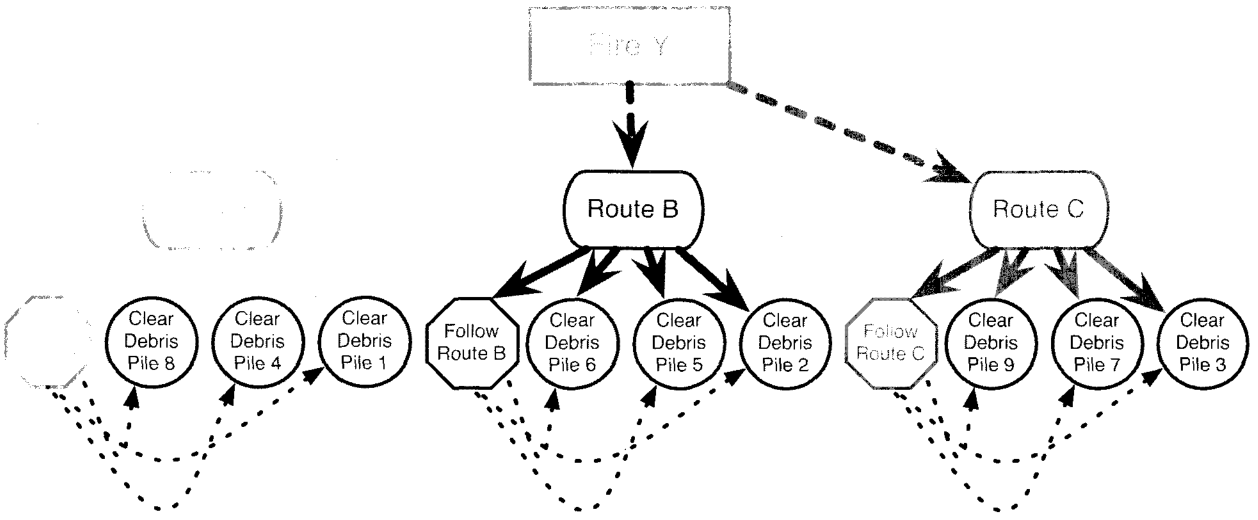


Figure 2.2: A hierarchical task network illustrating the decomposition of the plans for the fire truck to reach fire Y by each of the three routes shown in Figure 2.1. Large colored dashed lines indicate an OR relationship; solid lines indicate AND relationships; the small dotted black lines indicate a precedence dependency from the source to the target of the arrows.

allocations, the sub-network associated with assigning a particular fire to a particular task will be different for each fire truck, as the specification of the network is dependent on an agent's current location. Furthermore, the potential decompositions associated with assigning different tasks and routes to fire truck agents must be considered in tandem, as reliance on bulldozers serves to inter-constrain different fire trucks' schedules. This interdependence makes the problem more difficult for HTN solvers. To represent the space of possible plans in an HTN network would require an exhaustive enumeration of potential plans akin to the hierarchy for enumeration we present in Chapter 1.2.2. We aware of no work in the HTN literature that could tractably allocate and schedule tasks given a network which is super-exponential in the number of tasks and routes.

While the space of potential decompositions and networks is huge, the problem of searching the space of possible decompositions may be made easier by the fact that potential decompositions have distinct form and are highly inter-related. In the disaster response domain, for instance, inter-agent constraints take particular forms, rather than the full space of potential interconnections that may exist in supply chains or the COORDINATORS program. Furthermore, spatial structure serves to inter-relate potential decompositions - some routes may be clearly better than others, for instance. Similarly, spatial structure can suggest tasks for shared allocation, or limit the space of constraint allocations and routes considered.

Hierarchical task networks have the expressive capability to address inter-agent constraints in our domains of interest. However, approaches that address only how to schedule an existing network rather than how that network is formed solve only one part of the problem of determining solutions in domains of interest. Furthermore, computationally intensive approaches to determining quality schedules given a decomposition may lead to intractable performance for the large space of potential decompositions associated with problem instances in domains with intra-path constraints. Fast, time-extended approaches will certainly need to exploit domain structure to limit the space of routes, allocations, and sequences.

Multi-agent routing in domains with inter-agent constraints

Smith and Bullo address problem domains where the capabilities are divided among different agents, each of which can provide a distinct service (Smith *et al.*, 2009). In order to address particular tasks all services required by a task must simultaneously co-locate at the task site with a goal of minimizing expected delay. They propose three different policies - forming teams where all services are represented, forming different teams where the collective services are sufficient to represent particular task types, and a version of the task-specific teams policy that schedules intervals of times in which task-specific teams will address tasks of a given type. Given a set of assumptions, they prove bounds on solution quality in comparison to the optimum.

Zheng and Koenig formulate a time-extended approach to coordinating agents in a multi-robot domain with some independent tasks and some tasks with simultaneity requirements (Zheng & Koenig, 2008). Their approach is to initially allocate single agent tasks using a standard sequential single item auction, and then to allocate tasks with simultaneity requirements using reaction functions, which express the cost in distance terms of visiting a task with simultaneity requirements at different points in an agent's schedule. They use approximations so that the full space of potential sequences need not be considered. Their approach is exponential in the largest coalition required to address a task and polynomial in the number of agents and targets.

While both of these approaches can potentially offer insights into methods for determining coalition structure in domains with simultaneity constraints, in neither approach are there constraints that affect path-planning between tasks. Neither approach is sufficient for making routing decisions in domains where making routing decisions requires allocating and sequencing constraints.

2.3.3 Full horizon planning

We are aware of two approach styles that reason about the full space of potential coordination solutions and can optimally and provably solve the coordination problem in domains with intra-path constraints. The first method is to formulate the problem as a distributed POMDP. The second approach is to represent the full domain structure in a linear program and to solve the program using **mixed integer linear program (MILP)** solvers. In this section we examine both of these approach styles.

We first consider POMDP approaches. Static domain instances of our problem could potentially be represented as just Markov decision processes, but POMDP representations are of interest as they can account for uncertainty in transitions and observations. In general, distributed POMDP approaches are so computationally expensive as to be intractable for any but the smallest problem instances, but current research is addressing some of these limitations in order to expand the applicability of POMDP approaches. Of particular interest to this work are Kwak et al.'s work in exploiting coordination locales to gain tractability in POMDPs (Kwak *et al.*, 2009). As is the case in our domains of interest, Kwak et al. note that in many domains agents can often act independently even though interaction may need to occur at particular times and places; interaction may be required to avoid interference, or for task-level collaboration. The sites that may require interaction are referred to as coordination locales; for instance, a narrow corridor may represent a coordination locale, as agent interaction may be required if two agents attempt to use the corridor simultaneously. Task locations can also represented coordination locales. The approach they take involves doing most planning using relatively efficient single-agent POMDP solvers for most agent planning, and invoking a social model shaping approach to managing agent interactions. The social model shaping approach identifies points in the individually determined policies where agent interference means that agents following the policies would get lower than expected rewards due to interference, and points where one agent could help another agent and should receive higher reward. The shaping approach then manipulates agent rewards to induce reward-positive behavior and discourage reward-negative behavior. The approach yields better performance than an approach that does not reason about agent interactions, and achieves comparable performance and is much faster in domains with coordination locales than other state-of-the-art distributed POMDP techniques. This particular approach style will not guarantee optimal performance, but represents an important contribution in exploiting domain features in order to obtain good solutions.

There are two mathematical programming approaches that directly apply to our domains of interest. First, we consider Koes et al.'s work in multi-robot task allocation to a team of robots in the USARSim domain, a high fidelity simulated search and rescue environment also associated with a RoboCup competition (Koes *et al.*, 2005). In this domain tasks have associated rewards that decay linearly, and the system can be constrained in a variety of ways including task ordering, time-oriented task constraints, capability constraints, and global resource constraints. To enable their system to simultaneously optimize in the space of scheduling, task allocation, and path planning they use a mathematical programming approach. All task requirements, constraints, and robot path planning information are translated into mixed-integer linear inequalities. Once the problem has been formulated as an MILP it can be passed to a solver, which can determine the optimal assignment in terms of the objective of agents to tasks, including all agent schedules. The MILP runs in time exponential in the number of constraints. Koes et al. recognize that exponential time is often prohibitively expensive, but in many situations they can obtain reasonable solutions quickly by starting the solver with approximate solutions. They also note that the algorithm is any time, and thus can potentially produce reasonable solutions quickly that satisfy all constraints but may be non-optimal in terms of reward. If extra planning time is available, the solver can continue to run, improving the solution towards optimality gradually.

This approach could be used to determine optimal solutions in a somewhat simplified version of our domains of interest; however, for problem instances that involved more than a few agents and constraints the approach will be intractable, as the complexity is exponential in the the number of tasks and constraints. To illustrate these points, we consider how to represent the disaster response domain with intra-path precedence constraints in terms that would be compatible with Koes et al.'s approach. To do this we need to make one important assumption: all path distances are pre-computable. This assumption requires greater examination. Koes et al. note that mathematical programming techniques are poorly suited to long sequential plans like those required in path planning (Bererton, 2004). For this reason their approach pre-computes all path costs between task locations. In the disaster response domain this is problematic, as if we interpret fire tasks as the only tasks then we cannot pre-compute path costs, as the time taken to reach a fire task will be affected by intra-path constraints. We can accommodate this restriction by widening our interpretation of tasks to include all debris piles. In the disaster response domain, this would require the following:

1. Creating a task associated with each debris pile, requiring bulldozing capabilities.
2. Generating tasks along the roads to each side of the debris pile.
3. Generating precedence constraints that limit the ability of the fire trucks to pass through the debris area from either side until the debris pile task is addressed.

With these steps taken, path costs could be precomputed, as all precedence restrictions would be encoded in tasks and constraints. It is important to note in the Koes et al. approach tasks may go unassigned, so not all debris pile tasks would need to be allocated and scheduled in an optimal solution.

The approach taken by Koes et al. does not make any effort to exploit the particular features of domains with intra-path constraints - problem definitions are fed directly into an off-the-shelf MILP solver. A mathematical programming approach that attempts to exploit domain structure is the work of Mills-Tettey (Mills-Tettey, 2008). This approach focuses on domains with intra-path constraints as well as capacity constraints, choice of locations at which to perform actions, time constraints, and mutually exclusive resources. The approach takes a route-centric view of the problem, using a set-partition formulation to consider the space of possible routes and associating a set of side constraints with the formulation. The mathematical program is then solved using a branch-and-price approach. The approach is a bounded optimal anytime approach, potentially determining sub-optimal solutions quickly and providing a bound on the level of sub-optimality; if given sufficient run-time, it can determine provably optimal solutions to static domain instances.

Depending on the size of the domain, the level of uncertainty, and the availability of computation time these approaches may be appropriate for domains with intra-path constraints. Kwak et al.'s POMDP approach is an approximate algorithm and does not guarantee optimal results, but is applicable to domains with potentially high levels of uncertainty in perception and state transitions. Both of the mathematical programming approaches are capable of determining optimal results in static domain instances given sufficient computation time.

The main reason not to use these approaches are scalability and tractability. These approaches are all computationally expensive even for small domains, and as domain size increases the cost of the approaches can potentially increase exponentially. For domains with tens of agents, hundreds of tasks, hundreds of intra-path constraints, and large environments containing many potential routes optimal or near-optimal approaches are likely to require days, weeks, or months of computation time. Additionally, while the POMDP approach factors in uncertainty, the mathematical programming approaches will produce explicitly scheduled coordination results assuming static domain instances. If there is pronounced uncertainty in the environment, such as dynamic task issue, uncertain intra-path constraint locations, or significant duration uncertainty, the approaches must re-plan to ensure that solution quality remains high. For scenarios where computation time and tractability are significant concerns using these methods to determine provably optimal solutions will not be feasible. The mathematical programming approaches are anytime, however, meaning that in many cases well before they have produced a provably optimal solution they may produce a reasonable suboptimal solution; furthermore, they can produce a bound on the difference in quality between the produced solution and the optimal solution. This makes the mathematical programming approaches interesting points of comparison for faster approaches, as comparison could be done in terms of performance in a given amount of time for domain instances that are too large to be tractably solved optimally.

2.4 Multi-robot planning and execution

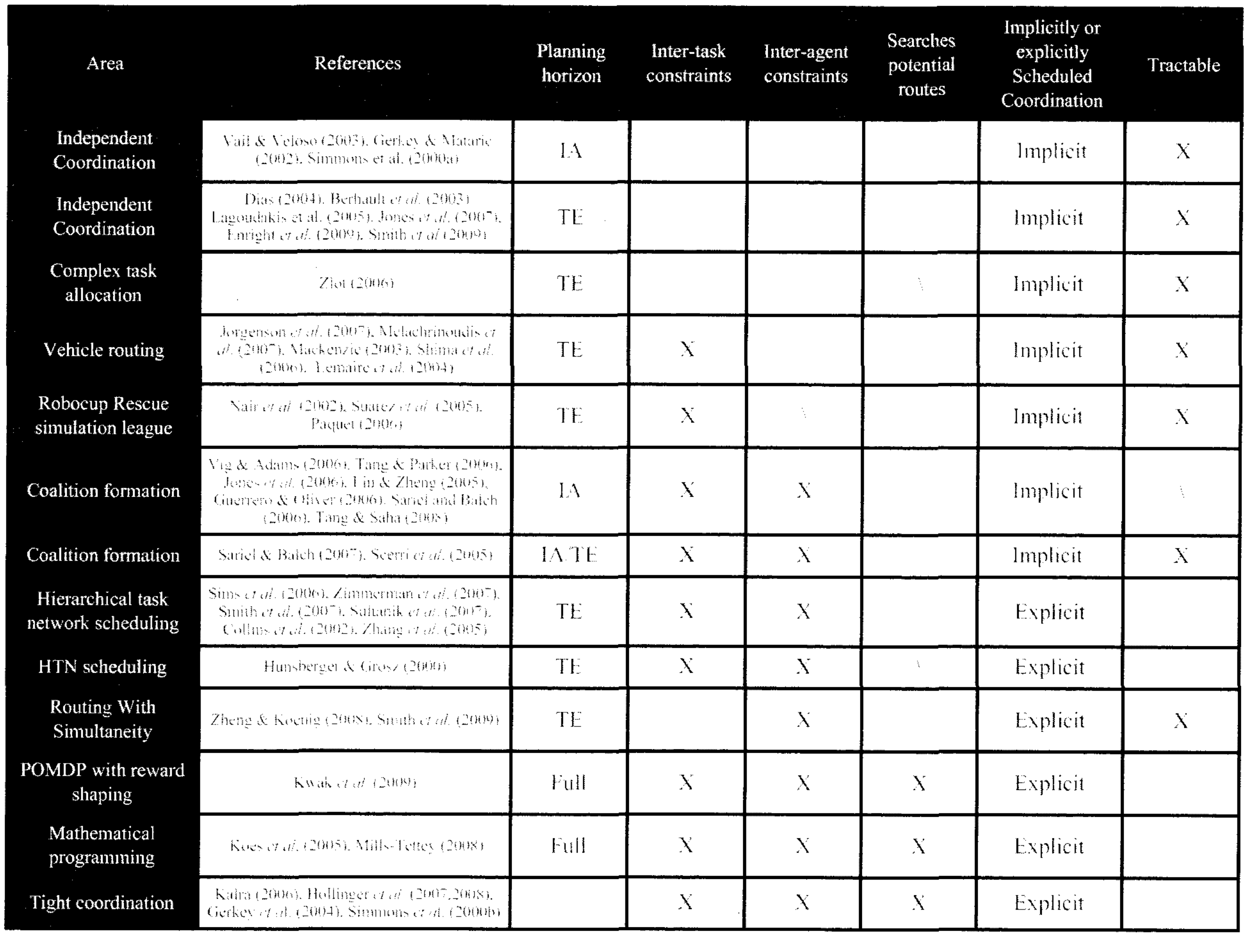
In this section we consider work in multi-robot coordination in which task allocation is not in question - allocations have been made, and robots have some set of goals to which they have been assigned. The question then becomes the manner in which the robots should move and perform actions to execute their goals.

One subset of multi-robot planning and execution work focuses on mitigating the interference effects associated with placing many robots in a domain. Robots operating in the same arena must not run into each other and should avoid situations where one robot delays another, by sitting idly in a doorway for instance. Bererton (Bererton, 2004) proposes a mathematical programming approach for optimally solving multi-agent path planning problems where interference can occur. Ryan (Ryan, 2007) provides a graph decomposition algorithm for road map planning where only a single agent can occupy a node at any given time. While we would ideally consider interference in our algorithms, our domains of interest make intentional interactions with other robots more important to performance.

Other work attempts to help agents coordinate actions in collaborative environments, where agents must combine efforts to address mutual goals. The MVERT framework, developed by Stroupe, uses a behavior-based control method where agents pick actions based on expected actions of other agents with a goal of maximizing team performance (Stroupe, 2003). Simmons et al. use a behavior-based controller with a distributed planning layer for coordinating a team of specialized robots performing autonomous construction tasks (Simmons *et al.*, 2000b). Schouwenaars et al. use an MILP-based formulation to solve trajectory generations problems of teams of helicopters moving through cluttered environments that must maintain communication ranges. Gerkey et al. and Hollinger et al. both address the problem of collaborative search for detecting invaders, where a team of agents must ensure that areas are cleared without allowing the invader to escape detection (Gerkey *et al.*, 2004) (Hollinger *et al.*, 2007) (Hollinger & Singh, 2008). Kalra uses a market-based approach called Hoplites to coordinate actions among teammates where tasks require tight coordination, where interaction between teammates must be constant to achieve a good solution (Kalra, 2006). While Kalra's approach would be sufficient to coordinate agents in our domains of interest once assignment, routing, and scheduling decisions had been made, in our domains teammates do not need to constantly interact to maintain good performance. In our domains of interest once tasks have been allocated and scheduled, if schedules go according to plan than agents may not need to interact at all - if a bulldozer clears debris on schedule then the fire truck is generally not affected by other bulldozer's actions. This independence of action except at the scheduling level means that Hoplites is not necessary in our domains of interest, as they do not require tight coordination.

2.5 Summary and discussion

In this section we presented work related to the problem of coordinating agents in domains with intra-path constraints. Our central contention is that no existing work addresses the problem of tractably determining high quality explicitly scheduled coordination solutions in domains with intra-path constraints. We summarize important features of related work in Table 2.1. We group references by general areas, and examine the potential to apply the approaches to our domains of interest. In order to produce explicitly scheduled coordination solutions an approach must have two primary attributes: the approach must be capable of reasoning about inter-agent constraints and must employ time-extended allocation. An approach that does not reason about inter-agent constraints cannot represent the relationships between task- and constraint-addressing agents, and an approach that does not reason about time-extended assignment cannot produce explicitly scheduled coordination in domains where some or all paths to routes require assigning more than one constraint-satisfaction duty to constraint-addressing agents. The first segment of work that has



Area	References	Planning horizon	Inter-task constraints	Inter-agent constraints	Searches potential routes	Implicitly or explicitly Scheduled Coordination	Tractable
Independent Coordination	Vail & Veloso (2003), Gerkey & Mataric (2002), Simmons et al. (2000a)	IA				Implicit	X
Independent Coordination	Dias (2004), Berhault *et al.* (2003) Lagoudakis et al. (2005), Jones *et al.* (2007), Enright *et al.* (2009), Smith *et al* (2009)	TE				Implicit	X
Complex task allocation	Zlot (2006)	TE			\	Implicit	X
Vehicle routing	Jorgenson *et al.* (2007), Melachrinoudis *et al.* (2007), Mackenzie (2003), Slima *et al.* (2006), Lemaire *et al.* (2004)	TE	X			Implicit	X
Robocup Rescue simulation league	Nair *et al.* (2002), Suarez *et al.* (2005), Paquet (2006)	TE	X	\		Implicit	X
Coalition formation	Vig & Adams (2006), Tang & Parker (2006), Jones *et al.* (2006), Lin & Zheng (2005), Guerrero & Oliver (2006), Sariel and Balch (2006), Tang & Saha (2008)	IA	X	X		Implicit	\
Coalition formation	Sariel & Balch (2007), Scerri *et al.* (2005)	IA,TE	X	X		Implicit	X
Hierarchical task network scheduling	Sims *et al.* (2006), Zimmerman *et al.* (2007), Smith *et al.* (2007), Saltanik *et al.* (2007), Collins *et al.* (2002), Zhang *et al.* (2005)	TE	X	X		Explicit	
HTN scheduling	Hunsberger & Grosz (2000)	TE	X	X	\	Explicit	
Routing With Simultaneity	Zheng & Koenig (2008), Smith *et al.* (2009)	TE		X		Explicit	X
POMDP with reward shaping	Kwak *et al* (2009)	Full	X	X	X	Explicit	
Mathematical programming	Koes *et al.* (2005), Mills-Tettey (2008)	Full	X	X	X	Explicit	
Tight coordination	Kalra (2006), Hollinger *et al* (2007,2008), Gerkey *et al.* (2004), Simmons *et al.* (2000b)		X	X	X	Explicit	

Table 2.1: Summary of related work in terms of the most important features in the context of this dissertation. The symbol X means that the given reference has the characteristic indicated by the column. The symbol \ means that the reference partially possesses the characteristic.

both of these characteristics is hierarchical task network scheduling. HTN techniques could be used for scheduling given constraint networks, and could produce explicitly scheduled coordination solutions; however, approaches to HTN scheduling contain no mechanism for reasoning about the space of multi-agent path plans that could be adopted in domains with intra-path constraints. Thus such approaches fail to meet our criteria that they provide a method for searching potential routes. Two categories of approaches, POMDPs and mathematical programming, can represent the full space of potential plans and potentially even determine optimal solutions, but have serious tractably concerns for any but the smallest problems.

While many approaches have the potential to produce implicitly scheduled coordination in our domains of interest, reasoning about explicit scheduling is necessary to accurately assess solution quality given the presence of intra-path constraints. The work that could be used to produce explicitly scheduled coordination either does not provide a methodology for searching the space of potential routes, or does provide a methodology but can only be employed in small domain instances due to exponential complexity concerns. Producing high quality explicitly scheduled coordination solutions quickly is a problem not fully addressed in the literature.

Chapter 3 Approach summaries/Minimum time approaches

In the previous chapter we considered existing approaches to coordination found in the literature and their potential for determining coordination solutions in our domains of interest. We determined that the approaches in the literature were not sufficient to address possible scenario requirements; particularly, that no existing approaches were capable of tractably determining high quality, time-extended explicitly coordinated solutions to domains with intra-path constraints. A central contribution in this work is the introduction of methods that can determine high quality solutions without requiring excessive time for computation. In the first section of this chapter we introduce our suite of approaches to coordination; our suite consists of five different approaches, each of which takes a somewhat different tact in searching the space of possible coordination solutions in domains with intra-path constraints, and has strengths and weaknesses when considered in the context of scenario requirements. We argue that collectively these approaches will serve to address a large space of potential scenario requirements. In the rest of this chapter we cover our first two approaches, which are the least computationally expensive of all of the approaches. Finally, we end the chapter with a summary and discussion.

3.1 Our approaches to coordination

In this section we introduce our approaches and talk about potential scenarios where they may be most appropriate for deployment. The most important features of the approaches are summarized in Table 3.1. In the last part of the section we consider the process of balancing scenario factors in selecting an approach and detail some of our own methods for approach evaluation.

Approach name	Task planning horizon	Constraint planning horizon	Searches potential routes	Implicitly or explicitly scheduled coordination
Independent tasks	IA	IA	No	Implicit
Allocate-then-coordinate (ATC)	IA	TE	After task allocation	Explicit
Tiered auctions with instantaneous task allocation (TA-IA)	IA	TE	During task allocation	Explicit
Tiered auctions with instantaneous task allocation (TA-TE)	TE	TE	During task allocation	Explicit
Genetic algorithms (GA)	Full	Full	During task allocation	Explicit

Table 3.1: The approaches to coordination presented in this work, which differ in terms of planning horizons, if and when potential routes are searched, and whether coordination is implicitly or explicitly scheduled.

3.1.1 Independent tasks

Approaches that explicitly scheduling coordination are likely to outperform implicitly scheduled approaches in many scenarios, and the bulk of the work in this thesis will address explicitly scheduled coordination. However, there are scenarios in which implicitly scheduled coordination may be the most appropriate solution technique, particularly when uncertainty is high or time for planning minimal. Furthermore, an implicitly scheduled approach can serve as a performance baseline for other approaches.

We call the implicitly scheduled coordination approach developed in this thesis *independent tasks*. While implicit coordination approaches could potentially consider a wide variety of factors in determining agent plans, our approach reasons only about easily determined factors as we intend this method to use a minimum of computation time. The approach uses a market-based technique inspired by independent coordination approaches to do instantaneous allocation of both tasks and constraints. Using an implicitly scheduling approach can potentially result in delays for agents, as they will simply wait until conditions are met so that they can continue to execute their plans. The approach we take does not explicitly account for inter-agent constraints in determining coordinated behavior, but is very fast and can potentially be very responsive to uncertainty. Consider, for instance, a disaster response scenario where a helicopter has spotted fire locations around a city but lacks the sensing capability to tell whether or not roads are passable. Doing long horizon planning for either fire trucks or bulldozers may be impossible, as the agents may not have the relevant data to produce accurate explicitly scheduled plans. In such a scenario, it may be best to have agents attend to whatever tasks or intra-path constraints are closest. Similarly, in domains where there is little time for planning, a short horizon solution based on an independent coordination assumption may be the best use of available computation resources.

3.1.2 Allocate-then-coordinate

For static domain instances, explicitly scheduled approaches are likely to outperform implicitly scheduled approaches, as explicitly scheduled approaches can accurately account for multi-agent constraints during planning. At the same time, searching the coordination space in domains with intra-path constraints is a challenging problem; to properly evaluate the effects of allocating a task to an agent requires a search of the space of possible routes and associated intra-path constraint assignments and sequences for those that will produce high quality. Determining an explicitly scheduled solution for each potential task allocation is likely to be a computationally expensive process. One way to mitigate this difficulty is to reason about only implicit scheduling during task allocation, but then to determine an explicitly scheduled solution for a particular task allocation strategy.

The **Allocate-then-coordinate (ATC)** approach adopts this methodology. Tasks are allocated using an allocation strategy similar to that used in the independent tasks approach. This approach differs from independent tasks in that once a task is allocated, task-addressing agents will search in the space of possible routes for a high-quality explicitly scheduled plan which accounts for intra-path constraints - this is the "coordinate" phase of the approach. To determine this explicitly scheduled solution requires search in the space of potential routes as well as making assignment and sequencing decisions for constraint-addressing agents.

Using the ATC method accounts for the two factors that will be most detrimental to solution quality if the independent tasks approach is followed: that agent schedules will not be explicitly coordinated, and that agents do not consider intra-path constraints when selecting routes. The ATC approach is designed to be only slightly more computationally expensive than independent tasks, to ensure that well-coordinated behavior is produced, and to account for the presence of intra-path constraints during route selection. The biggest deficiency of the approach is that these factors are not accounted for during task allocation. In some domains, the allocation step may not be of great importance to solution quality. Consider a scenario in the on-order manufacturing domain where orders tend to be of uniform importance and involve roughly similar plans; in such a domain the exact orders assigned to agents may be relatively unimportant compared to doing a good job of routing and respecting intra-path constraints. The ATC approach is appropriate in scenarios where planning time is still scant but not so minimal as to preclude using any approach but independent tasks. The approach may also be useful in domains where the level of uncertainty is not so high as to preclude explicit scheduling, but high enough to where solutions must be regularly re-evaluated.

In our implementation of ATC we use a market-based method to allocate tasks to agents; agents bid for tasks based on shortest path distance without accounting for intra-path constraints. Once assigned a task an agent will do a search through the space of possible routes to the task. The route search involves holding a series of single item sequential auctions targeted at agents that can address intra-path constraints along those routes - these agents bid for tasks based on adding tasks to the ends of their schedules. The auctioning agent will select the highest quality route to the task, assigning intra-path constraint satisfaction duties along those routes to the agents that won tasks in the auctions. By using bounding techniques during the route search process the time required to search the route space is kept to a minimum.

3.1.3 Tiered auctions with instantaneous task assignment

The ATC approach supports the generation of explicitly scheduled plans after allocation, but does not consider the results of route search to determine explicitly scheduled plans during task allocation. Our next incremental expansion of the search in the coordination space is to consider the results of route search during allocation. Considering route search during allocation should result in solutions that better account for the limited resources available to address intra-path constraints. This sort of reasoning should be especially beneficial in domains with a high density of intra-path constraints. If we consider the disaster response domain, there may be domain instances where reaching most fires requires negotiating a long series of debris piles; in this scenario bulldozer availability may have a great deal of impact on which fires should be assigned to fire trucks. Fires that may be relatively unimportant but that can be reached by routes that require negotiating few or no debris piles become likely candidates for assignment. Similarly, in the on-order manufacturing domain it may be the case that a number of orders have arrived that require a similar set of parts, meaning long delays for acquiring the parts from tower agents. In this scenario assigning orders that do not require those parts is likely to lead to better solutions.

While considering route search during allocation has the promise of providing greater solution quality, the size of the search space being considered grows substantially; searching this space will almost certainly lead to greater computation cost. To keep the search tractable it becomes important to use heuristics and bounding techniques to focus the search whenever possible.

Combining route search with allocation requires that agents reason about each other during the allocation phase; standard market-based frameworks contain no provision for agents to solicit the participation of other agents during the bidding process associated with allocation. To support this reasoning we developed a novel method, tiered auctions, that allows agents to consult with other agents during the bid formation process. In our implementation agents conduct route search in order to bid on tasks; searching the space of routes involves holding sub-auctions to assign intra-path constraints to constraint-addressing agents. We maintain a short horizon by only using instantaneous assignment for task-addressing agents, though constraint-addressing agents may be assigned to more than one constraint. We call this approach **tiered auctions with instantaneous task assignment (TA-IA)**.

3.1.4 Tiered auctions with time-extended task assignment

Considering coordination during allocation can lead to high quality solutions, but considering only a short time horizon still confines the search to a relatively small region of the potential coordination space. Our next approach, **tiered auctions with time-extended task assignment (TA-TE)**, extends the time horizon for search to include multi-task assignments while maintaining the consideration of multi-agent coordination requirements during allocation. In order to maintain tractability this approach style will not consider every possible multi-task assignment and does not use a fixed time horizon; instead the approach depends on spatial heuristics to group tasks together that should potentially be jointly allocated. In the disaster response domain, for instance, fires can be grouped that are spatially near each other and separated by few debris piles, as a route that brings an agent close to one fire may allow nearby fires to be easily accessed. In many domains the structure of multi-agent constraints makes spatial reasoning powerful, and we believe that employing spatial heuristics may allow for tractable long time horizon search in many domains with intra-path constraints.

Our methods for long time horizon consider multi-task allocation by grouping tasks geographically for joint allocation, and by searching for appropriate multi-task allocations during the route search process; we enhance our tiered auction method using both of these heuristics for grouping tasks. By using spatial heuristics and bounding techniques we can extend the planning horizon without exponentially increasing the time requirements for search.

While extending the planning horizon can potentially increase solution quality in many domains, in others planning far into the future may be of limited utility. Successful execution of long-term plans is unlikely in domains with substantial uncertainty, and if long-term plans must be constantly revised it may be a better use of available time to focus on short-time horizon planning. Furthermore, in many domains there may not be time available for long-time horizon planning even if uncertainty levels are relatively low. Finally, greedy strategies like the ones that we adopt for long-time horizon search may lead to plans where particular agents monopolize available resources for addressing intra-path constraints.

3.1.5 Genetic algorithms for full coordination space search

In each of the previous methods we limit the scope of the search - by using a short time horizon, not fully considering multi-agent constraints, or using greedy independent strategies. In our final approach style we focus on methods to search in the full coordination space; this requires employing an indefinite time horizon, fully accounting for the effects of multi-agent constraints, and optimizing not the performance of any particular agent but all agents in concert. In this approach style we represent full solutions in the coordination space: allocations, sequences, and routes for agents. Solutions are evaluated using simulation, which can account for the real effects of adopting interdependent agent schedules. Search occurs by preferentially selecting good solutions and randomly altering those solutions; over time, better and better solutions should emerge.

Such an approach should be applicable in any domain with intra-path constraints independent of the density or kind of intra-path constraints; however, finding good solutions in domains with high constraint density or difficult constraints may require even more time for search. The approach can be non-heuristic, depending on randomness and selection to guide search over time, or employ heuristics that try to modify solutions beneficially to improve the search more quickly. While this approach can offer superior performance by searching more widely in the search space and accounting more accurately for multi-agent constraints, it has a number of requirements. First, resources and time for computation must be plentiful; if little time is available it is almost certain that one of the more heuristic approaches will determine better solutions than those produced in the same amount of time by evolutionary search. Second, enough information to accurately represent and simulate the environment must be accumulated at a central location; once accumulated, much of the computation can be parallelized, but this approach offers poor distributability. Third, for domains with high uncertainty searching with an indefinite time horizon almost certainly represents some wasted effort.

Our implementation of full coordination space search uses **genetic algorithms (GAs)** as our randomized search algorithm. For a broad introduction to genetic algorithms see Goldberg (1989). Each genome represents a full coordination solution. Crossover and mutation operators randomly alter agent allocations, sequences, and routes. Some operators can use heuristics to improve solutions. Different selection functions are used to focus the search while maintaining solution diversity.

3.1.6 Approach selection and evaluation

During the rest of the thesis we will give experimentally validated insight into how each of these approaches performs in domains with different factors in terms of domain characteristics and levels of uncertainty. Selecting an approach for a given domain will be a process of determining scenario requirements, detailing resources available for the search for a coordination solution, assessing probable domain characteristics, and considering what uncertainty may exist in the domain. Each scenario will be slightly different, and there is the potential that an approach will require parameter tuning or other minor adaptations to achieve maximum performance in different scenario variations.

An essential factor in approach selection is evaluation of performance in different domain instances. We primarily evaluate the approaches within a simulated disaster response domain. In terms of domain representation, we model fire trucks as operating on a graph-based map of the road network. Bulldozers are not restricted to this road-network, and instead operate on an occupancy grid representation of the world; they must avoid buildings but can otherwise move freely between debris pile locations. Figure 3.1 shows a screen shot from our simulator. We use a hybrid event and discrete time simulation method for determining the effects of solution execution.

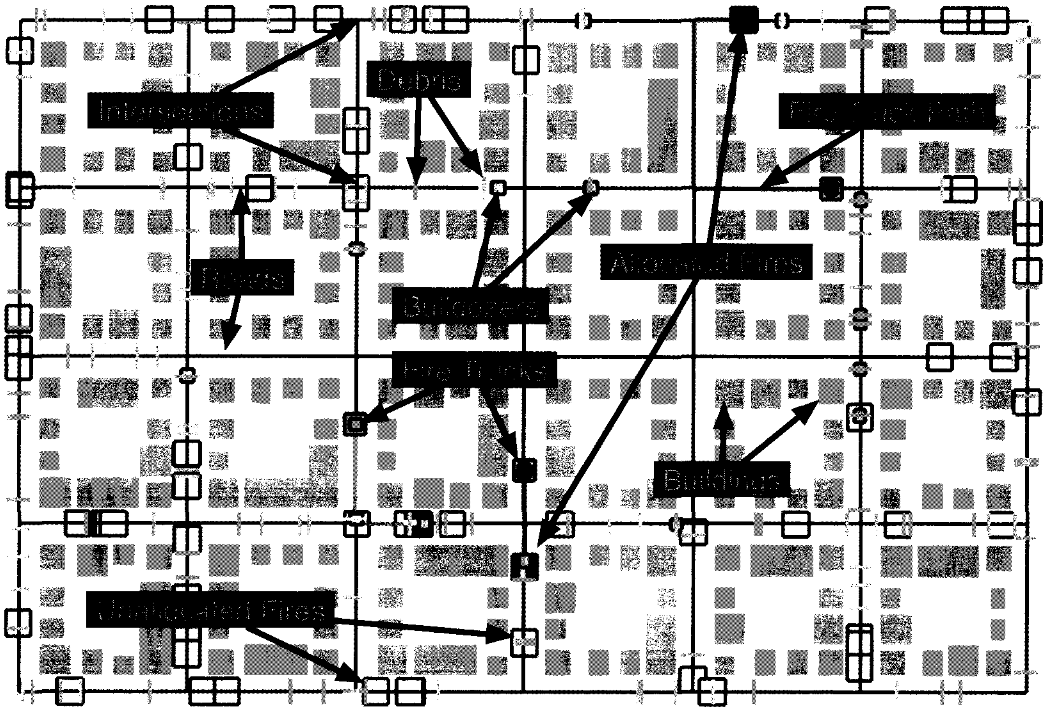


Figure 3.1: A screen shot from our simulation visualizer

In this and subsequent chapters each of the approaches is presented as running synchronously within a single thread of computation. This allows us to present the coordinated search in terms of function calls, whereas in a real deployed system the coordination would almost certainly be asynchronous and require network communication. In each of the approaches we will draw attention to when the algorithms could and could not be run in parallel on a number of different machines.

Within this evaluation framework we track two primary metrics for solution performance - the quality of the solution in terms of the objective function, which in the disaster response domain is the sum of values of affected buildings, and the total computation time necessary to compute the solution. Detailed evaluation in terms of other metrics that capture different aspects of scenario requirements is left for future work.

We initially explore approach performance in static domain instances. In the simulated disaster response domain this means that the environment map is fully known, including intra-path constraint locations and characteristics, that task issue is static and task characteristics including objective value and decay, locations, and required durations are known precisely, and that agents can reliably communicate. In static domain instances in terms of solution quality and computation time we would expect a general trend: searching more comprehensively in the space of potential solutions should yield better coordination solutions but require more search time. More comprehensive searching can include using explicitly instead of implicitly scheduled coordination, accurately accounting for inter-agent constraints during allocation and sequencing, or extending the planning horizon. While we expect to see some increases in time associated with more comprehensive search, we will attempt to limit those increases in developing heuristics and bounding methods that exploit domain features in order to speed search while maintaining high quality.

A final observation is that the trends described in the previous paragraph are significantly complicated by the presence of uncertainty in the environment. In Chapter 6 we consider how different kinds of uncertainty can be accommodated by the approaches and how they alter the trade-offs between using comprehensive search and reacting quickly to new information.

3.2 Minimum time approach introduction

Having introduced our approaches to coordination we move on to detail our first two approaches, which we group under the category of minimum time approaches. These approaches are designed to be as fast as possible for use in domains where there is very little time available for computation.

In this chapter we focus on our two least expensive approaches from a computation perspective. Both use a short time horizon for planning, and neither consider intra-path constraints when making task allocation decisions. These short time horizon approaches focus only on the portion of the coordination space which involve the next few actions each agent will perform. Reasoning over only a short time horizon can potentially be detrimental in terms of solution quality, as the future consequences of actions do not factor into determining coordination solutions. At the same time, there are a number of reasons that a short-time horizon approach may be appropriate for a given scenario. Some scenarios may allow for virtually no planning time - in such a scenarios concentrating the search on the next few actions each agent will perform may be far more effective than trying to do a more limited search with a longer time horizon. Other scenarios may involve high levels of uncertainty which render long horizon planning ineffective. In such scenarios it can actually be preferable to use short horizon planning even in terms of solution quality, as a greedier, shorter horizon approach can outperform a long horizon plan that incorrectly judges the future state of the world.

In this section we explore two different short horizon approach styles that differ primarily in their treatment of multi-agent constraints during the coordination search process. The first method, independent tasks, is an approach that uses implicit scheduling for coordination, treating task and constraint assignments as separate independent coordination problems during coordination search.

The second approach, Allocate-then-coordinate (ATC), allocates tasks without considering inter-agent constraints, but then uses a process called route search to determine high quality explicitly scheduled coordination solutions to allow task-addressing agents to quickly reach task locations. This process involves considering different routes along which task-addressing agents could reach tasks and determining constraint assignments and sequencing for constraints which must be negotiated along those routes. We describe the general method the approaches take in searching the coordination space as well as the specifics of our implementations of the approaches within the disaster response domain. We illustrate the performance of the approaches within a simulated disaster response domain.

In the next section we introduce some background information in market-based coordination methods. We then detail the independent tasks and ATC approaches, evaluating the approaches in a simulated disaster response domain. Finally, we summarize and discuss the chapter's findings.

3.3 Background

Before we begin to describe the details of our approach we need to provide some background information on market-based approaches to coordination. In this section, we focus on the sequential single-item auction, a widely used technique in domains with independent coordination. Sequential single-item auctions focus on determining coordination solutions quickly in a parallelized or distributed fashion, to quickly re-plan in response to new environmental data, and to be robust to single points of failure (Dias *et al.*, 2004a). In this section we describe the approach as used in our own work in a disaster response domain where fire trucks have full terrainability and no intra-path constraint are present (Jones *et al.*, 2007). The goal in this domain is similar to that in our own disaster response domain - to maximize the sum of affected building values.

Market-based systems rely on auction mechanisms for task allocation and on individual agent planning for sequencing in time-extended solutions. While some implementations allow for many different agents to hold auctions, in our system we used a single auctioneer - an agent acting as a dispatcher in the domain. The **Dispatcher/Auctioneer (D/A)** holds auctions when it is informed of newly discovered fires. The D/A determines an ordering for newly discovered fires and holds an auction for the first fire in the ordering. The goal for the auction is to assign the task to the agent that can gain the most reward from addressing the task given that each agent will already have a set of assigned tasks it must address. The D/A forms an auction call for this task containing information like the fire's location and the value function. When the fire trucks receive the auction call they must determine a sequence consisting of both the fires to which they are already committed and the new fire which maximizes the overall scheduled reward it expects to result from addressing all tasks. As the domain involves only independent coordination, fire trucks can individually search the space of potential task sequences without considering the actions of other agents. We use an exhaustive approach to sequencing for small schedules, and for larger schedules we adopt a simulated annealing optimization approach. Fire trucks bid for the fire by computing the added value the agent expects to receive for addressing the task given that it already is committed to addressing other tasks. We call this **marginal reward** - the extra reward achieved by incorporating the task into an existing schedule.

Once bids have been submitted the D/A then determines a winner for the task auction, a process called clearing the auction. Our clearing mechanism consists of determining which bid for the task is highest - this is the bid corresponding to the agent that can gain the most in terms of reward for incorporating the task into its existing schedule. The D/A informs the winning agent of the assignment, and the task will be incorporated into that agent's schedule. If other tasks remain unassigned the D/A will hold new auction rounds for those tasks; agents will bid in subsequent auctions based on the results of previous auctions. The sequential single-item auction format means that only a single task is offered for auction at a time, but a sequence of tasks may be offered in different auctions.

Sequential single-item auctions generally produce reasonable solutions, and have been shown to have bounds on performance quality for some domains and objective functions (Lagoudakis *et al.*, 2004). Additionally, for distributed implementations they are compact in terms of communication, as they do not require sharing large maps or other environmental data; instead, auction calls and single value bids are the only necessary communication. Auctions also serve to distribute or parallelize much of the most expensive computation, the individual agent path-planning and task sequencing. While partially centralized, as bids generally must be returned to a single central auctioneer figure, market-based implementations can be robust to single points of failure, as any agent in the domain can act as an auctioneer. By periodically running auctions to allocate newly discovered tasks domains with dynamic task issue can be accommodated; by equipping agents to hold re-auctions the discovery of new environmental information that effects the quality of the solution can be accommodated. In practice market-based methods have been shown to be a resilient and efficient way to coordinate distributed teams of robots (Dias *et al.*, 2004a).

While they have a variety of strengths, sequential single-item auctions do not necessarily produce optimal solutions in independent coordination domains. One potential source of inefficiency is in the sequencing algorithms used by agents to form bids. For many objective functions finding an optimal sequence is an NP-Hard problem that is some version of the Traveling Salesman problem. Nothing precludes an optimal sequencing approach to be used within a sequential single-item auction, but to avoid exponential complexity many implementations will avoid optimal approaches and instead use faster sub-optimal approaches. The second source of inefficiency is that the sequential single-item auction method only searches a small portion of the space of possible time-extended allocations. By allocating only a single task during an auction, some multi-task allocations that would improve the solution are not considered.

3.4 The independent tasks approach

Having presented background in sequential single-item auctions we move to our first approach, independent tasks. As described in Chapter 1, one primary difference between approaches is whether or not they explicitly or implicitly schedule agent interactions. In an implicitly scheduled approach, the specific times that tasks or constraints will be addressed are not scheduled in advance. Instead, agents attempt to execute their plans, and if prevented from doing so due to intra-path constraints will wait until those constraints are met. The focus in implicit scheduling approaches is determining coordination solutions that will tend to result in reasonable performance even without explicitly scheduling interactions. Implicitly scheduled coordination approaches can potentially be quite sophisticated, factoring in detailed domain knowledge. Our independent tasks approach is optimized to require minimum computation time but still determine reasonable solutions, though the performance may potentially steeply degrade for domains with high intra-path constraint density. This is intended to be the approach with the minimum requirement in terms of computation time, best suited for domains where there is little time for computation or high levels of uncertainty.

The approach we take in our independent tasks approach is to ignore inter-agent constraints in determining agent plans, and to treat the task and constraint assignment problems as separate independent problems; we allocate tasks and constraints using a simple market-based approach for single task allocation. The central design question in the independent tasks approach is by what method we determine task and constraint-satisfaction allocations and plans given that the requirements of other agents will not factor into the planning. For task allocation, we could allocate tasks on the basis of maximizing estimated single task reward, but this may not be the best strategy. Any estimation of reward that agents expect to obtain from addressing a task is likely to be highly inaccurate, as it will not account for the requirement that constraints will generally need to be satisfied on the potential routes to the tasks. Using optimistic estimates may result in especially poor performance, as task-addressing agents may try to reach tasks that are physically distant, requiring negotiating large numbers of constraints.

In our approach we assign task-addressing agents to the tasks that are estimated to be addressable in the shortest amount of time. In this approach agents are likely to be assigned tasks that are physically proximate to their current locations; the routes to these tasks are relatively unlikely to be blocked by large numbers of obstacles. Constraint-addressing agents are allocated constraint-satisfaction duties using a similar approach. A more sophisticated implicit coordination approach could attempt to generally reason about which constraints should be cleared - potentially having agents travel longer distances to clear debris piles that block particular thoroughfares or prevent access to otherwise navigable areas - we leave exploration of such techniques as future work. The approach of just having agents clear constraints that are near to them and require little time to address should result in little wasted effort.

While using an implicit coordination approach does not preclude reasoning over a longer time horizon, we believe doing so is unlikely to substantially improve performance and will likely cost more in terms of computation. If more time is available for computation, using an approach that better accounts for multi-agent constraints in constructing explicitly scheduled solutions is likely to be a better use of the computation time than reasoning about how to potentially assign agents a series of tasks. Another avenue for improvement could be to attempt some re-planning when agents discover that they will be delayed in reaching their assigned tasks - trying to take a different route or address a different task. If the re-planning does not account for the presence of inter-agent constraints, the newly determined plan is unlikely to improve on the existing plan, and re-planning frequently will substantially increase the computational cost of the approach.

3.4.1 Independent tasks implementation

The independent tasks approach uses a market-based method for determining allocation of both tasks and constraints-satisfaction duties. Our approach is detailed in Algorithm 1. We use an auction process that is a slight variation from the sequential single item auction approach presented in Chapter 3.3. A list of unassigned, unaddressed fires or debris piles - henceforth both called tasks - is generated by the D/A. The list of available tasks is transmitted to idle agents - agents that are not currently involved in addressing other tasks. Idle agents determine the task that is nearest to them in terms of Euclidean or Manhattan distance and place a bid for that task consisting of the distance. For efficiency purposes, these bids represent only distance estimates for bulldozers and fire trucks, as doing full path plans for each agent to reach each task can be computationally costly for domains with many agents, tasks, and constraints. The auction is cleared as follows: if two or more agents have the same closest task the agent with the lowest bid is assigned the task, breaking ties using lexicographic order; the other agents that bid on the task are added to a list of agents that need to form new bids. If an agent is the sole bidder for a task it is assigned the task. All losing bidders from a round of bids are sent a new list of unassigned tasks and asked again for bids. The process continues until all agents have been assigned a task or no tasks are left to assign.

Once agents have been assigned tasks, they determine schedules to complete the tasks. Bulldozers run optimal single agent path planners in the grid space. Fire trucks use A* to determine the shortest path, which may require passing any number of debris piles. If fire trucks encounter debris piles on their routes they do not attempt to find alternate routes to the fire; instead, they wait until either the debris pile is cleared and they can pass or the fire completely destroys the building, meaning that no value can be gained by addressing the fire. Bulldozers become idle once they have addressed their assigned debris pile, and fire trucks become idle once they have extinguished their assigned fire or their assigned building is completely destroyed. Idle agents will bid for new tasks whenever the next auction cycle is conducted.

Algorithm 1: Independent task allocation for fires to fire trucks and debris piles to bulldozers in the disaster response domain

```
Input: T: List of unassigned, unaddressed tasks
       A: List of idle agents
Result: Single task assignments for each available agent
1 while A is not empty do
2     foreach a in A do
3         bid ← BidOnClosestTask(a,T)
          // Bids contain a task bid.task, bidding agent bid.agent, and a distance
             estimate bid.value
4         Add bid to bidList
5     foreach bid in bidList do
6         if IsLowestForTask(bid,bidList) then
              // If bid represents the lowest bid for the particular task, award
                 it
7             Assign bid.agent to bid.task
8             Remove bid.agent from A
9             Remove bid.task from T
```

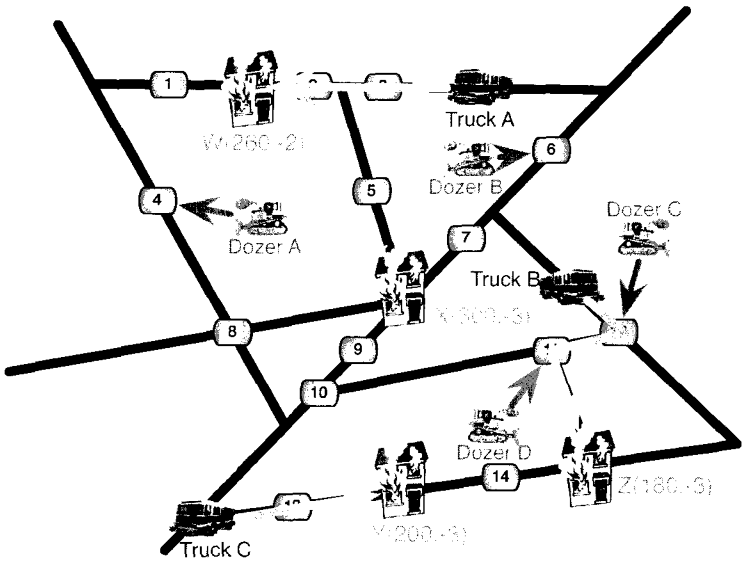


Figure 3.2: The results of independent task allocation in an instance of disaster response. Arrows represent the routes that agents will take to reach their assigned task or constraint. The numbers in parentheses on fire labels represent the current reward and linear decay of the indicated fire. Greyed ovals are debris piles, and are numbered for reference.

In Figure 3.2 we show the allocation that would result from following this method for initial assignment in an example domain instance. We make a few observations about this example. First, in some cases this approach can lead to reasonable coordination - Fire Truck B should be able to easily reach its assigned fire along its shortest path route as both constraints are scheduled for immediate clearing. The use of implicitly scheduled coordination can hinder task completion in some cases, however. Both Trucks C and A will be forced to wait for debris piles to be cleared for potentially long periods, as no bulldozers are scheduled to address the constraints which lie on the agents shortest paths. Bulldozers A and B will instead address debris piles 4 and 6, which do not lie on any truck's intended path. As the density of intra-path constraints increase, situations where implicit coordination finds good solutions are likely to get more and more infrequent, resulting in potentially long delays in task completion.

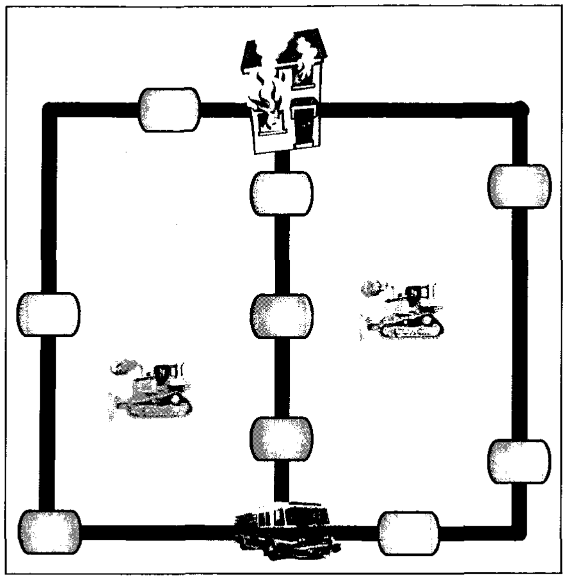
While using our implicit coordination approach does not necessary result in maximally efficient behavior, it represents an extremely fast way to determine coordination solutions. We will present quantitative results supporting this assertion in Section 3.6.

3.5 Allocate-then-coordinate

Our next approach style, allocate then coordinate (ATC), is designed to remedy a central deficiency in solutions that result from using an independent task approach: using an implicit scheduling approach may result in unnecessary delays in task completion and consequently lower performance. The ATC approach uses explicit scheduling in determining plans for agents, ensuring that schedules are accurate and that delays are minimized. The ATC approach is the least computationally expensive of our explicit scheduling approaches. It accomplishes the goal of being computationally inexpensive by using a similar strategy to that used in the independent tasks approach to do task allocation; once allocation is performed, however, the approach reasons about the space of possible routes and allocation and sequencing of constraint along those routes, using explicit scheduling to accurately determine the quality associated with different coordination plans. This approach separates allocation from coordination, reasoning about coordination only after allocation has been performed. Neglecting the results of route search and explicit scheduling during task allocation can have potentially detrimental effects on solution quality, but limits the required computation time.

From an algorithmic perspective the focus of the ATC approach is in the process of determining the route that should be taken to an assigned fire given the requirements imposed by intra-path constraints. This algorithm requires methods to generate potential routes and to determine route quality given the availability of other agents to address intra-path constraints. To maintain tractability, bounding methods must be used to limit the space of potential routes, as there can be an extensive group of potential non-repeating paths through the node graph that can be taken to reach a target node.

Explicitly scheduling coordinated plans to allow tasks to be addressed may require that a somewhat longer planning horizon be considered for some agents. Consider an example from the


(a) The example problem instance

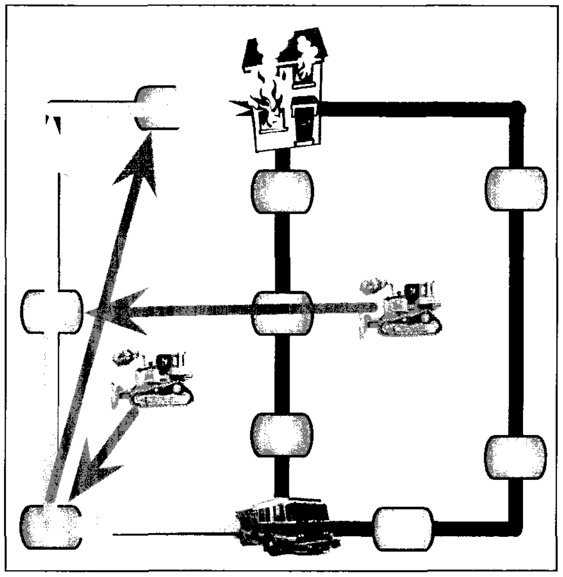

(b) Results of the earliest completion heuristic for constraint-satisfaction duty assignment and sequencing given the fire truck route shown in blue. Bulldozer plans are shown as green arrows.

Figure 3.3: Example in the disaster response domain in which to fully determine an explicit schedule to allow the fire truck to reach the fire requires assigning more than a single constraint satisfaction duty to some constraint-addressing agent

disaster response domain, shown in Figure 3.3. Suppose that the fire truck has been assigned the indicated fire and must now plan a route to reach that fire. If we were to limit the planning horizon to assigning only a single constraint per constraint-addressing agent, there would be no way given this set of agents and routes to construct an explicitly scheduled solution for the fire truck to reach the fire, as some debris pile would go unassigned and unscheduled. In the ATC approach we extend the planning horizon, but only for constraint-addressing agents. Extending the planning horizon means that sequencing decisions become a part of the search space for these agents, potentially substantially increasing the search space over an instantaneous allocation approach. One approach we can take to mitigate the difficulties in searching this larger space is to develop heuristics to guide the search for constraint assignment and allocation during route search. We developed a simple heuristic that drastically reduces the search time associated with assigning and sequencing constraints during route search: the earliest completion time heuristic. The heuristic is based on the intuition that addressing constraints earlier in a route as quickly as possible is a good approach to allowing fast passage. The earliest completion time heuristic considers constraints in the order the task-addressing will reach them on a particular route; the constraints are assigned to the constraint-addressing agent with the earliest completion time for the constraint given that agents may already have assignments. In terms of sequencing, the only considered sequences are those where constraint-addressing agents add tasks to the ends of their schedules. The results of using the heuristic in the example shown in Figure 3.3(a) for a particular fire truck path are shown in Figure 3.3(b). This heuristic may not produce the optimal assignment and sequencing for constraints on a given route, but it produces reasonable solutions quickly; we believe that considering a wider array of potential constraint assignments and sequences is unlikely to substantially improve performance and would increase the computational cost of route search.

As we stated in the independent tasks section, determining the basis on which agents will calculate bids for tasks is an important feature of the independent tasks approach. In the independent tasks approach we awarded fires on the basis of distance. While this is an effective metric in the independent task approach, it does not account for the fact that different tasks can have differing importance. Addressing a task that is very close to a truck may be a good strategy for implicitly scheduled approaches, however addressing a higher reward task instead of a lower reward task is generally a better strategy, even if the higher reward task is slightly further away. Moving towards an approach that attempts to maximize reward rather than minimizing distance can improve performance.

While rewards should factor into allocation decisions, allocating based solely on maximizing the expected reward for assignments can lead to inefficient allocations. An approach that allocates tasks based on maximizing single task reward for agents will allocate tasks that are relatively far away from agents even if the added gain over addressing a nearby task is minimal. While that may be appropriate if the goal was to maximize single task rewards, it is not well aligned with the goal of maximizing the total reward received over all addressed tasks. In the disaster response domain making efficient use of time is paramount, as rewards for all tasks are decaying constantly; while agents will not be explicitly reasoning about future tasks, we can design a **bidding function** where agents attempt to get as much reward as they can as quickly as they can. The approach we experimentally determined yielded the best performance is an average reward strategy – the amount of reward the agent will receive for the task given its scheduled completion time divided by the total duration of time it will take to reach the task and address it. Conducting auctions based on maximizing average reward serves to make agents greedier about obtaining reward quickly and then moving on to address other tasks, striking a balance between minimizing distance and maximizing single task reward.

In the next section, we first detail the basics of route search, where a single fire truck has been assigned a fire and must search through the space of possible routes to determine the fastest way to reach the fire. We then discuss how the route search process as used in the ATC approach.

3.5.1 Route search specifics

Our approach to route search is designed to determine a schedule that will allow a fire truck to reach a fire as quickly as possible while using minimal computation; determining an accurate schedule requires planning a route to the fire and allocating and scheduling debris clearing requirements that lie along the route. By using bounding and heuristics we find a fast route while exploring only a fraction of the space of potential routes and associated debris clearing assignments and sequences.

Our method for conducting route search depends on one primary insight in order to tractably search the space of possible routes: the primary objective of the route search is to determine the fastest route to the target task. If at any point it is determined that a route or set of routes cannot possibly represent a faster path to a fire than another route that has already been explored then that route or those routes need not be considered. We structure the search to quickly find a fast route and then use that route to bound the search space.

The route search procedure from the fire truck's perspective involves three major sub-components:

1. A single agent planner capable of producing unique routes to reach the goal
2. A function that can take a route and turn it into a schedule
3. A procedure for taking a route, allocating and assigning intra-path constraints to other agents, and updating the schedule based on the results

We consider each of these sub-components in turn. For the first sub-component, the single agent planner we use is a version of A* that finds routes in the fire truck road network. The A* algorithm is initialized with the domain map, starting and goal locations, and an additional parameter that represents an upper bound for the number of intra-path constraints that are permitted to lie along the route - this is used in the full algorithm as described later. Once initialized, the A* algorithm will produce non-repeating paths that do not require passage through more than the allowed number of intra-path constraints in order of total distance. The `GetNextShortestPath` call can be used to produce the next path. The algorithm functions by maintaining the search queue between calls to `GetNextShortestPath` - subsequent calls will cause the search to continue until the next fastest path to reach the target is determined.

The A* algorithm produces a sequence of nodes, but does not associate arrival or departure times with the nodes. Our second sub-component is a function that takes the sequence of nodes and turns it into a schedule, including scheduled arrival and departure times at all nodes in the route. In this domain there are three types of nodes - intersections, tasks, and constraints. We assume that intersections can be negotiated without delay. Tasks that a task-addressing has not been assigned and does not intend to stop to address can also be negotiated with no delay. The task that is the goal of route search will be the final node in the schedule, with the completion time assumed to be the sum of the arrival time at the fire and the required duration for the truck to extinguish the fire. The final type of node are constraint nodes. In terms of scheduling unaddressed constraints can be separated into two categories - those that already have been assigned to some agent, potentially as an assignment associated with some other route plan, and those that are currently unassigned. In our system, constraint commitments and scheduled completion times are broadcast after they have been determined, so this information can be used to populate the schedule without contacting other agents. In terms of scheduling, it will either be the case that a constraint is scheduled to be cleared by the time the truck will arrive at the site, causing no delay, or that the scheduled completion time is after the truck's arrival at the node. In this case the departure time from the node will be the time at which the constraint is addressed; the time spent waiting for the constraint to be addressed represents a delay in the truck's schedule. During the initial process of determining a schedule from a node sequence any unassigned debris piles are assumed to have zero delay, though this will frequently be an optimistic assumption.

The final building block of the algorithm is the method for taking a route with unassigned constraints and determining an accurate, explicitly coordinated schedule for reaching the target task along this route. We use the earliest completion time constraint assignment heuristic, implementing the heuristic using a sequential single-item auction. The order of the auction is determined by the order in which constraints will be reached along the route. This auction has one important difference than the auction used in the independent tasks approach. In that auction once an agent had been awarded a task in the auction it was assumed that the task was the agent's responsibility and it could begin execution. In the auctions associated with route search, the route represents only one possible plan among many that will allow the truck to reach the target task; even if an agent is assigned a constraint to address in the auction it will generally only represent a portion of search, and not a command to execute. At the same time, agents cannot simply disregard assignments, as those assignments must be reflected in future bids for constraints that occur later in the route. We developed a simple method for distinguishing to the agents bidding in an auction on what basis they should be determining a bid - each route a fire truck can take to a task is distinguished by a unique **auction context**. Sending a context as part of a constraint auction call allows bulldozers to regard awards in the correct light, leading to accurate bidding for future tasks.

Algorithm 2: `AllocateUnassignedConstraintsInRoute` - assigns bulldozers to unassigned constraint-satisfaction duties in a schedule using the earliest completion time heuristic

```
Input: a: agent associated with route
       s: current route schedule
       bestTime: current completion time associated with fastest route
       D: List of bulldozers
Output: Updated schedule s with allocations for each unassigned constraint along the
        schedule, or a schedule with potentially unassigned constraints that takes longer
        than bestTime to complete
1  context ← GenerateUniqueContext()
2  while there are still unallocated constraints in s do
3      c ← FindNextUnassignedConstraint(s)
4      call ← ComposeAuctionCall(c,context)
5      Initialize winningBid.time to high value
6      foreach bull in D do
7          bid ←BidOnConstraint(bull,call) // Bids contain bidding agent bid.agent, a
               constraint bid.constraint, and a completion time bid.time
8          if bid.time < winningBid.time then
9              winningBid ← bid
10     MakeConditionalAward(winningBid.agent,call)
11     s ← UpdateScheduleWithWinningBid(winningBid.constraint,winningBid.time)
12     if s.compTime > bestTime then
           // Delay associated with constraint completion means that this route no
               longer can improve on bestTime
13         return s
14 return s
```

The route allocation process is shown in Algorithm 2. The fire truck acts as an auctioneer, and when auctioning unassigned debris piles along a new route forms an auction context for that route - the auction context indicates that this auction should be associated with a particular plan for an agent to reach a fire. An auction call is formed for an unassigned debris pile consisting of all relevant information as well as the auction context. This is sent to bulldozer agents. When a bulldozer agent receives an auction call it checks the auction context against an internal database associated with the auctions in which the bulldozer is currently involved. If the auction context is new, the bulldozer creates a new entry in the database consisting of a copy of its current schedule, which could include other constraint commitments - this is its **context schedule**. The bulldozer then forms a bid for the task based on adding the task to the end of its context schedule - in our system we do not allow schedule reorganization. In this version of the disaster response domain a bulldozer can independently determine a completion time for the task by using a single agent planner to determine a path from the last location on its context schedule to the debris pile, and sum the arrival time with the required duration to clear the debris pile. As the auction is intended to implement the earliest completion time heuristic, bulldozers bid completion times. This newly created schedule including the auctioned debris pile is stored as a potential context schedule by each bulldozer. Once the fire truck auctioneer has received the bids it clears the auction by conditionally awarding the task to the lowest bidder, representing the bulldozer that can address the constraint the most quickly. The truck sends an award message to this agent indicating that it has be assigned the task in the context auction. The winning bulldozer then adopts the potential context schedule as its new context schedule - all future context bids will be placed using the context schedule that includes the task. The fire truck can then continue to auction the debris pile that lie along the route.

For each awarded and scheduled constraint the fire truck can update its schedule to account for the completion time. If the fire truck is not scheduled to reach the debris pile site by the scheduled completion time, no change is necessary as the schedule already reflects zero delay. If, however, the fire truck will reach the debris pile site before scheduled completion, then it introduces a schedule delay that is the difference between the original scheduled departure from the debris pile location and the current scheduled completion time for the constraint - the time at which passage will be allowed. This delay must be added to all arrival and departure times for subsequent nodes in the schedule with one exception - if there was already a scheduled delay associated with an assigned constraint later in the schedule then that later delay may be reduced or eliminated by delay earlier in the schedule. Once all unassigned debris piles in a route have been assigned and scheduled the fire truck should have an accurate estimate for completion time for the fire that will result by taking this route. Other routes can then be considered.

Algorithm 3: RouteSearchATC - route search algorithm used in the ATC approach for a fire truck trying to determine the fastest route to single a target fire

```
Input: a: Agent that needs to route search
       f: Target fire
       C: List of currently assigned constraint satisfaction duties and their scheduled
          completion times
       B: List of bulldozers
       m: Graph map of environment
Result: Single best route bestSched for agent a, which will have associated bulldozer
        assignments and schedules as well
for numCons ← 0 to NUM_DEBRIS_CONS do
    InitializeAstar(m, a, f, numCons) // Initialize A* algorithm from a to f
        with constraint limit numCons
    while true do
        p ← GetNextShortestRoute()
        if p is not set then
            // No more unique routes with this number of constraints.
            break
        s ← GenerateScheduleFromRoute(p,C) // We need to pass in current
            commitments and the route
        if bestTime < DetermineZeroDelayCompletionTime(s) then
            // All remaining routes with this number of constraints cannot
                improve on bestTime even assuming zero delay
            break
        if bestSched is set and s.compTime < bestTime then
            // New schedule potentially improves on existing best route
            s ← AuctionUnassignedConstraintsInRoute(a,s,bestTime,B)
            if s.compTime < bestTime then
                // s improves on bestTime even after explicit scheduling of
                    unassigned constraints
                bestSched ← s
                bestTime ← s.compTime
```

We use each of these components in the algorithm for route search, as shown in Algorithm 3. There are a few important structural notes to point out about the basic algorithm. First, we search paths in order of the number of unaddressed constraints that lie along those paths, and then among routes that have the same number of debris piles by distance. We believe this to be a more effective strategy than searching strictly in order of distance for several reasons. For one thing, as we discuss below, the process of determining the quality of a route blocked by a number of debris piles is laborious and will generally require the involvement of other agents. If a route is blocked by no or only a few debris piles then its quality can be quickly determined and can be used in future bounding. Also, even if routes with few debris piles are much longer in distance than some other routes that are blocked with many debris piles, in domain instances with many debris piles, where debris piles take a long time to clear, or where there are few bulldozers it is unlikely that short paths blocked by many debris piles are going to allow fast passage to a fire.

The second aspect to note is that bounding is used to exclude entire sets of routes. First, we use the parameter NUM_DEBRIS_CONS to limit the number of allowable unassigned debris piles that can lie on any route to a fire. It is possible that by excluding some routes with many constraints routes that could actually represent faster ways to reach the fire will go unconsidered. This will be rare in most domain instances, however, as bulldozer resources tend to be in great demand. Additionally, even if a route that requires addressing a number of constraints may represent the fastest path to a task, it may be counterproductive to allow single agents to monopolize constraint-satisfying resources in order to reach a fire. We discuss this in more length in Chapter 4, as it is an even more important consideration in longer horizon approaches. The second bounding method that can exclude sets of routes serves to end the route search. Suppose that some route to a fire has been fully allocated – all debris piles along the route have been assigned and scheduled – and that the resulting completion time for the fire is *bestTime*. If, even assuming zero delay at unassigned debris piles, all remaining potential paths with a given number of debris piles are long enough in length that they cannot possibly allow the fire truck to extinguish the fire by *bestTime* then the search for paths with that number of debris piles or fewer can terminate - no remaining paths can improve on *bestTime*. By constantly updating *bestTime* with the best solution determined thus far in practice we need consider only a fraction of potential non-repeating routes.

This bounding method can also serve to disqualify routes during the auction process. For a route to be considered viable for auction it must be the case that assuming zero delay at unassigned debris piles the route could potentially improve on *bestTime*. As the auction proceeds, however, any delay that occurs at unassigned debris piles may make the route slower than bestTime. If this occurs the auction process need not continue, as the route being considered is not the fastest route to the fire - this check occurs in line 12 of Algorithm 3. We can also use similar reasoning to disqualify entire sets of potential routes. In Line 9 of algorithm 3 we do a check if the schedule generated from the route improves on the best time assuming zero delay at all constraints, whether previously assigned or unassigned. If this check fails, it means that purely in terms of travel time the routes that the A* algorithm is producing cannot possibly improve on the current best schedule. Thus no further routes with the indicated number of constraints need be considered.

The route search concludes when there are no viable routes with fewer than NUM_DEBRIS_-CONS to consider - the route associated with bestTime can be adopted. The fire truck can then send a new kind of award to bulldozers that have been assigned debris piles associated with the winning auction context - when this award has been received bulldozers can replace their current schedules with their associated auction context schedules. In our system they also broadcast their new commitments to all other fire trucks.

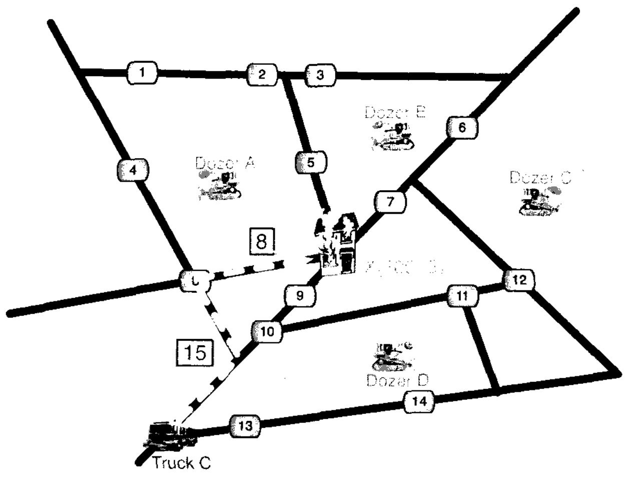


Figure 3.4: Example route search scenario in a disaster response domain. Fire Truck C has been assigned Fire X and searches the space of possible routes to reach the fire. The route being considered is shown as a blue dotted line, and the purple boxes show the length of time it will take the fire truck to navigate the indicated road segments.

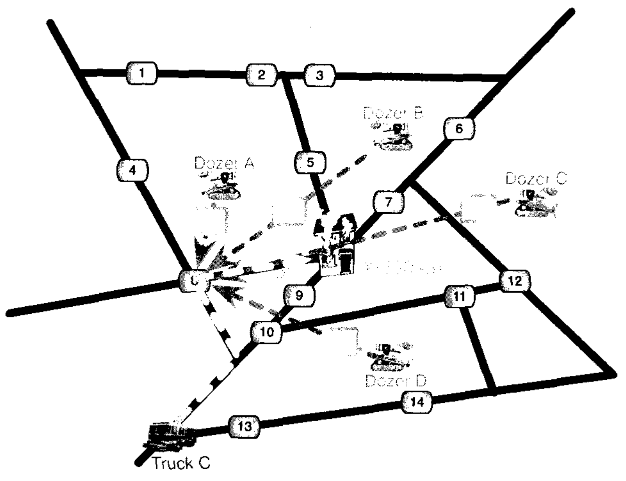


Figure 3.5: The fire truck assigns the context 10001 to the auction for this route and sends out a call for bids to the bulldozers. Bulldozers bid completion times given that they currently have empty schedules.

We now illustrate the route search process in an example, shown in Figures 3.4 to 3.9. In this scenario fire truck C has been assigned to fire X and enters into a route search loop to attempt to find the fastest route to the fire. The first viable path produced by the call to `GetNextShortestRoute` will be the route shown in Figure 3.4, which is the shortest path blocked by a single debris pile. Taking this route will require 23 cycles of travel time for the truck - 15 cycles to reach the debris pile, and 8 cycles to navigate from the debris pile to the fire. If fire X takes 4 cycles to clear once reached, then the completion time for the fire if there is no delay at the debris pile will be 27 cycles. To determine an actual completion time for the route the fire truck must assign and schedule the single debris pile on the route. The truck generates the auction context 10001 and sends an auction call for the debris pile with the generated context tag. In this example bulldozers have not yet been assigned to clear any constraints and bid their completion times for the debris pile based on empty schedules. Bulldozer bids are illustrated in Figure 3.5. The bulldozer that can clear the debris pile the most quickly is Dozer A, and it is awarded the clearing duty, as shown in Figure 3.6. Dozer A will not be able to clear the debris pile by the time the truck reaches the pile location, however, with a three cycle difference between the clearing time and the truck's arrival. This produces a three cycle delay in the task completion time. Taking this route results in a completion duration of 30 cycles, with an expected reward of 210 for task completion.

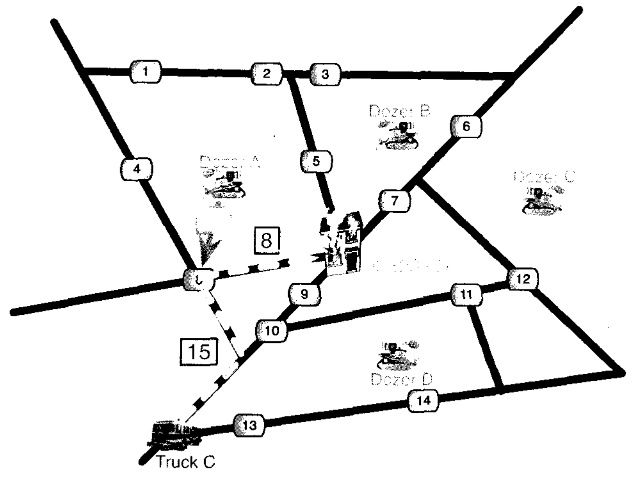


Figure 3.6: The fire truck assigns the debris pile to the earliest bidder, Dozer A, and can determine an accurate completion time for the task.

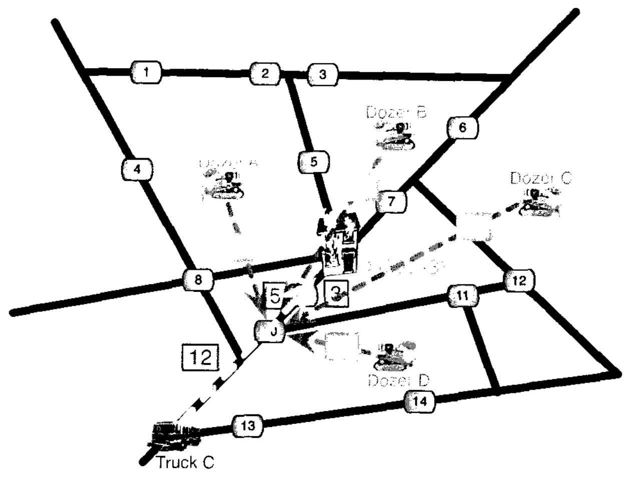


Figure 3.7: The truck considers another route to the fire, shown as a dotted blue line. The truck auctions debris piles for the routes using a new associated context, 10002. Bulldozer completion times for the first debris pile are shown.

The fire truck then must consider other routes to reach the fire. No other routes blocked by a single intra-path constraint exist, but there is a path with two debris piles, shown in Figure 3.7. If the truck were to take this route and the debris can be cleared without delaying the truck it will it will reach the fire in 20 cycles and extinguish it in 24; extinguishing the fire in 24 cycles improves on the current *bestTime* of 30 cycles. The truck associates a new context, 10002, with the route and sends out a call for bids for the first debris pile in the path. All bulldozers bid based on empty schedules, as there is a new context associated with the auction. The earliest completion time, 16 cycles, is bid by bulldozer D, which is awarded the task in the context auction. This completion time represents a delay of 4 cycles, delaying the minimum completion time for this route by 4 cycles to 28 cycles, which is still an improvement on a 30 cycle completion time associated with the first route. The truck then offers the next debris pile clearing responsibility with the same context, as shown in Figure 3.8. This means that Dozer D bids given that it must first clear its assigned debris pile before addressing the new pile, as it already has a scheduled assignment in its context schedule. Given that D has an existing commitment, Dozer A has an earlier completion time of 24 and is awarded the clearing duty. Factoring in the earlier delay at the first constraint site the fire truck will reach the second debris pile after 21 cycles. The truck will need to wait until 24 cycles to pass through the second debris pile site, representing an additional delay of 3 cycles. This means that the completion time will be 31 cycles, and this route is inferior to the first route. The fire truck can safely discard the route; if it were the case that unallocated debris lay further in the route they need not be auctioned, as the route cannot be the fastest route to the fire.

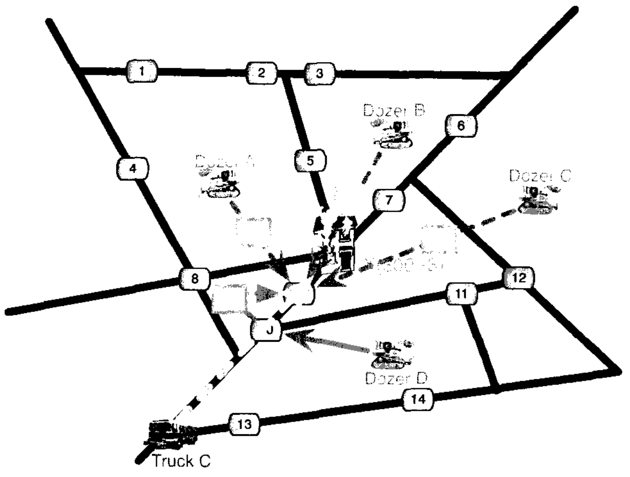


Figure 3.8: Bulldozer bids in terms of completion times for the second debris pile encountered in the route. Bulldozer D bids based on needing to address its assigned debris pile earlier in the route.

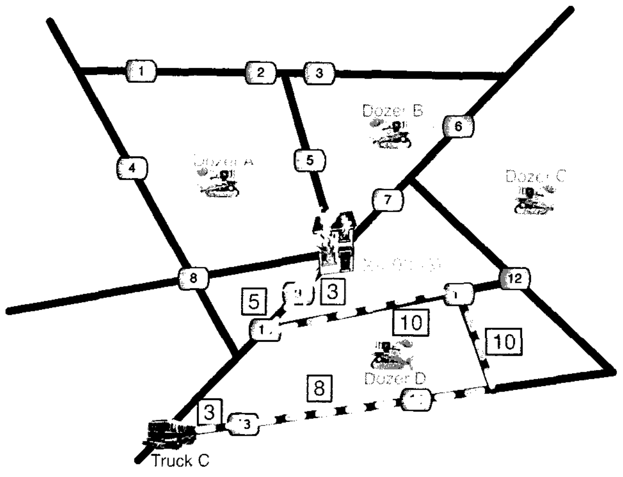


Figure 3.9: The shortest five-constraint route to the fire. The time required to traverse segments are shown in purple boxes.

A* would then produce the shortest five constraint path, as no paths with three or four debris piles exist. This path is shown in Figure 3.9. Even assuming no delay this route cannot result in the fire being completed in less than 43 cycles, much worse than the current best of 30 cycles. This route as well as other routes with five debris piles need not be considered. Similar logic means that higher debris pile routes need not be considered. The route search process can then conclude, and the schedules associated with the best context - 10001 - adopted.

3.5.2 ATC and route search

In the ATC approach route search occurs only after a truck has been assigned a fire. Fires are allocated in a market-based process that largely mirrors that used in an independent tasks approach: trucks bid for tasks relying on shortest path distance estimates for determining how quickly a fire can be reached. Such estimates will generally be optimistic. In our approach we move away from assigning tasks on the basis of distance and instead use the estimated average reward that the fire truck can obtain for the fire based on taking the shortest path and assuming zero delay. In each auction cycle all idle fire trucks submit bids for fires based on shortest path distance. The auctioneer selects the single highest bid among all bids; the fire associated with this bid is assigned to the bidding agent. Once an agent has been assigned a fire it uses the route search algorithm to determine how best to reach its assigned fire. If some fire trucks remain idle then a new auction round is held. New auctions are held periodically to re-assign idle agents - the frequency with which auctions are held is a parameter in the system. Holding frequent auctions may result in extra communication load, but infrequently holding auctions may mean that agents spend more time idly waiting for new assignments.

For static domain instances, the ATC approach should determine substantially better solutions than an independent tasks approach, as the selected routes are associated with explicitly scheduled coordination that not only accounts for intra-path constraints but does so during the process of route search. The ATC approach can take somewhat more time to determine solutions, however, as the route search process can be time-consuming even when using bounding and heuristics. One deficiency in terms of the time required for coordination is that the route search process cannot easily be conducted in parallel. If two fire trucks were independently and simultaneously conducting auctions as part of route search, bulldozers would not correctly be accounting for assignments associated with one agent in the auctions for another agent, and conflicts could result. Our method is to ensure that route search occurs serially - after task allocation the winning fire truck does route search, and only after a route has been determined will another truck be allowed to conduct route search. While this does not increase total computation time, it does have the effect of reducing parallelizability.

Thus far we have only considered assigning constraints to constraint-addressing agents during the route-search sub-auction process. It may be the case, however, that not all constraint-addressing agents are assigned to constraints as part of the route search process. In terms of overall system efficiency having bulldozers waiting idly to be tasked is likely to be inefficient. Thus for untasked bulldozers we implemented an Idle auction. Bulldozers that have not been actively tasked will be assigned intra-path constraints in an auction of the form used in the independent tasks approach. Being assigned a task in this auction does not preclude bulldozers from bidding in future route search auctions - the bulldozers will bid in these auctions based on empty context schedules.

3.6 Experiments and evaluation

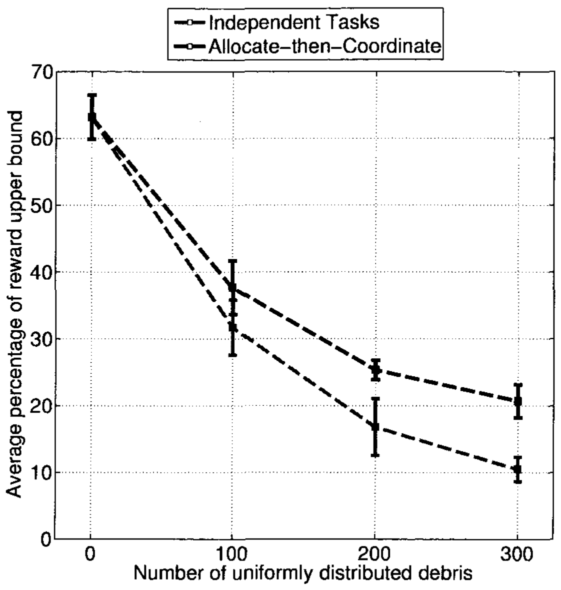
Evaluation would ideally involve comparing the performance of approaches by evaluating coordinating solutions generated for all possible domain instances. Given that this is not feasible, our evaluation process is to randomly generate domain instances with characteristics as specified by a number of different parameters. By randomly generating domain instances we approximate some of the different sorts of situations approaches will encounter. We then coordinate agents in each domain instance by using each of the different approaches in separate simulations. By comparing the results of coordination over a number of different randomly generated domains we try to approximate the true performance differences that result from using the different approaches. By changing the parameters that govern the random generation of domain instances we can then get an idea how the approaches cope with different kinds of domains - domains where intra-path density tends to be high, for instance, or where it takes a good deal of time to address tasks.

In this section we will examine how both independent tasks and ATC approaches perform in domains with varying intra-path constraint density. To that end we set our fixed domain parameters as follows: we run all experiments in the seven-by-five road network shown in Figure 3.1 with a team consisting of three fire trucks and 12 bulldozers. In each different domain instance agent locations are generated randomly within the central area of the domain environment. 100 fires are randomly scattered on the road network, with placement drawn from a uniform distribution. We use a linear model for reward decay, where the value for addressing a fire decays by a certain value for each cycle that the fire goes unaddressed. When the decay causes the fire value to reach zero no reward is obtained from addressing the fire. The value function for the individual fires is drawn from two independent normal distributions, one for initial value and one for value decay. We use the distribution $N(3000, 250)$ for initial value and $N(25, 5)$ units/cycle for decay. This means that a fire with average task value will reach zero after 120 cycles. Fires have zero duration, meaning that trucks extinguish them instantly on arrival. Debris piles require 10 cycles of bulldozer effort to clear.

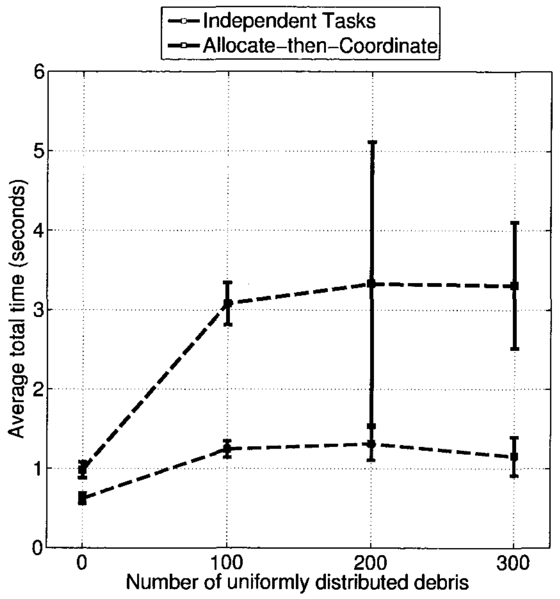
In these experiments we vary intra-path density, changing the number of debris piles that are present in the domain. In our experiments we set four debris frequency levels of 0, 100, 200, 300; at the 300 debris level this translates to an average of over five debris per road segment. In all cases debris piles are randomly scattered on the road network as drawn from a uniform distribution. For each debris frequency level we randomly generated and tested on ten different domain instances. Though domain instances have the same number of debris piles, the placement of debris piles, as well as the spatial relationship of the debris piles, the agent starting locations, and the fire locations all factor into the achievable rewards in the system.

We illustrate solution quality in Figure 3.10(a), which shows approach reward in terms of a proportion of the absolute upper bound on available reward. In this case the upper bound represents the summed initial value of all buildings at the time they were affected by the fire - this is the maximum reward available in the domain. As the density of intra-path constraints increases the performance of both approaches suffers. For all the domain settings with debris piles, however, the ATC approach outperforms the independent task approach. The performance gap widens at higher debris pile levels, as the independent tasks approach tends to produce solutions that are increasingly poorly coordinated. At the 300 debris pile level the average reward received by the ATC approach almost doubles that received by the independent tasks approach.

Figure 3.10(b) shows that even with the extensive bounding done during route search shows that there is an added computational cost associated with using the ATC approach. In domain instances with debris piles, the ATC approach uses an average of 2.6 times more computation than is required in the independent tasks approach. In some domain instances, route searches may occur that are especially costly in terms of computation time, leading to much higher standard deviations in the ATC approach. The computation requirements remain quite minimal for both approaches however, taking substantially less than 10 seconds to do all necessary computation.



(a) Average reward achieved by approaches as a proportion of an upper bound of available reward, where the bound is the summed reward of all issued tasks if they were addressed immediately. Standard deviations are shown as error bars.



(b) Average total time taken to produce coordination solutions. Trials were all run in a single thread and allowed 100% of a CPU on a quad-core Intel Xeon 3.8 GHz CPU.

Figure 3.10: The performance of the two minimum time approaches in a simulated disaster response domain

3.7 Summary and discussion

In the first part of the chapter we introduced our approaches for coordination in domains with intra-path constraints and discussed the general problem of approach selection and evaluation. Our different approaches each have strengths and weaknesses in terms of different scenario requirements, and determining which approach best fits a particular scenario will require reasoning about scenario requirements, resources, and domain features.

In the rest of the chapter we explored two approaches to coordinating agents in domains with intra-path constraints that are both capable of determining coordination solutions while using little computation time. Our first technique, independent tasks, treats the problem of coordinating task-addressing and constraint-addressing agents as two separate independent tasks allocation problems. This approach assigns tasks and constraints using a market-based approach where the goal is minimize distance for single task or constraint assignments. Task-addressing agents take the shortest path route to reach their assigned fires without consider how constraints may affect those routes. The result is implicitly scheduled coordination, as completion times for the tasks cannot be accurately set as all constraints along routes may not be allocated or scheduled.

Approach	High constraint density	Little time for planning
Ind. Tasks	–	++
ATC		+

Table 3.2: General guidelines on the strengths and weaknesses of the approaches in static domain instances. Strengths are denoted with the + symbol, and weaknesses with the – symbol, with particular strengths or weaknesses indicated by doubling these labels.

The other approach we considered in this section is the ATC approach. This approach does not substantially differ from the independent task approach in terms of task allocation, as tasks are allocated using a market-based approach without considering constraint allocation and scheduling. After task allocation has occurred, however, the algorithm enters a coordination phase, and searches the space of possible routes. Assessing routes requires determining allocations and sequences for constraint-addressing agents; this is accomplished using a series of sequential single item auctions. Rather than fully searching the space of potential constraint allocations and sequences, the powerful earliest completion time heuristic is adopted. An important component of making route search computationally inexpensive are the bounding methods which control the space of potential routes considered. The coordination phase is used to produce accurate, explicitly scheduled coordination in static domain instances. The use of the route search coordination phase in determining routes and constraint allocations and sequences results in substantially improved performance over the independent tasks approach, especially as intra-path constraint density increases. The coordination phase does introduce added requirements in terms of computation time, though computation time requirements remain minimal.

In Table 3.2 we summarize the findings presented in this chapter in terms of the relative strengths and weaknesses of approaches. We showed that the performance gap between independent tasks and ATC increases as the density of intra-path constraints increases. Thus the independent tasks approach makes a relatively poor choice if a given scenario has a high density of intra-path constraints. In terms of scenario requirements the primary strength of both of these approaches is that they determine solutions quickly. Each approach ignores substantial elements of the full coordination search space. The independent tasks approach ignores multi-agent constraints altogether when determining the coordination solution, and the ATC approach ignores constraints during allocation. For scenarios where there is scant time for determining solutions, however, the approaches each act reasonably, and obtain a substantial portion of the available reward.

The ATC approach represents our first explicitly scheduled approach, though explicitly scheduled coordination does not factor into allocation decisions. We will explore other explicitly scheduled coordination approaches in later chapters. The independent tasks approach is the only implicitly scheduled coordination approach we present during the course of this thesis. The independent tasks approach is a relatively simple version of implicit scheduling, and could potentially be improved in a number of ways. Additional factors could be incorporated into the heuristics used to determine coordination solutions; for instance, reasoning could include some notion of how addressing particular constraints will improve access to the domain environment. Extending the time horizon for planning is another potential avenue for improving performance. For constraint-addressing agents, extending the planning horizon could potentially be more useful. For instance, we could conduct a time-extended sequential single-item auction to potentially reduce travel time and speed up constraint satisfaction. While this would be a reasonable avenue for exploration, such an approach would substantially increase the computational requirement for the approach. If extra time for search is available, using a more computationally expensive explicit scheduling approach may improve performance more than extending the planning horizon.

Chapter 4 Tiered auction approaches

In the previous chapter we detailed two approaches to determining coordination solutions in domains with intra-path constraints that have minimal requirements in terms of computation time. The independent tasks approach is an implicitly scheduled approach, using auctions for single task and constraint assignments made without consideration of multi-agent constraints. The ATC approach produces explicitly scheduled coordination solutions, determining the fastest routes that task-addressing agents can take to assigned fires and corresponding schedules for constraint-addressing agents. Neither approach, however, considers the results of route search during the allocation process; instead, allocation decisions are made without considering the presence of intra-path constraints or the availability of agents to address those intra-path constraints. If these factors are not considered during task allocation then poor outcomes may result, especially in domains with a high density of intra-path constraints. In the ATC approach, an allocation decision may assign an agent a task that can only be reached by negotiating many intra-path constraints. If resources to satisfy those constraints are otherwise occupied, substantial delays can result. In this section we present approaches that explicitly reason about routing and intra-path constraint decisions during the allocation process.

An approach that considers coordination during allocation must perform route search during the allocation process. In a market-based system this means that route search must occur during the process of bid formation, as allocation decisions are made on the basis of agent bids. Standard-market based mechanisms contain no mechanism for soliciting the involvement of other agents during bid formation. We introduced one potential market-based approach that does permit such reasoning in our previous work in a domain with tight coordination requirements (Jones *et al.*, 2006). In this approach agents respond to auction calls by first selecting a tightly coordinated strategy for addressing the task specified in the auction call. These plans generally require the joint participation of a number of different agents filling roles in the plan, where the requirements to fill a role are possessing a certain kind of capability. Agents then allocate the roles using sub-auctions, where the agents would solicit the involvement of other agents in their plans. Only when roles are allocated and costs considered can agents place bids for the original auction. This system was fully implemented and has been used to coordinate a team of Pioneer and Segway RMPs working together with humans in a Treasure Hunt domain (Jones *et al.*, 2006).

While domains with intra-path constraints do not require tight coordination, the tiered auction mechanism can provide a method for allowing task-addressing agents to form accurate bids for tasks given that forming such bids requires soliciting the involvement of other agents. The approach we take is to have task-addressing agents hold route search sub-auctions in response to task auctions; bids are submitted based on the explicitly scheduled coordination solutions determined during route search. Task allocation decisions can then be made by the auctioneer based on accurate bids rather than the estimated bids associated with the approaches presented in the previous chapter. The tiered auction process allows for searching the space of allocations while taking into account required coordination, representing a more comprehensive search of the coordination space.

In this chapter we discuss two tiered auction approaches. The first is a short horizon approach that reasons only about single task allocations. Even though this approach is relatively short in horizon, considering allocation in tandem with the route search process makes adapting bounding and heuristics important for maintaining tractability. In the second approach we extend the tiered auction approach to multi-task, time-extended allocation. To negotiate the exponentially larger space of potential multi-task assignments we depend on heuristics that exploit domain spatial structure to group together tasks that can be jointly allocated to improve performance. In this chapter we first describe the two tiered auction approaches and then evaluate the relative performance of each approach versus the minimum time approaches presented in the previous chapter.

4.1 Tiered auctions with instantaneous task assignment

Using tiered auctions allows coordination solutions that consider the results of route search during the allocation process. While making allocation decisions that accurately account for intra-path constraints should result in higher quality solutions, even for short horizon approaches the portion of the search space considered is greatly increased. Even when using the earliest completion time heuristic for constraint assignment and sequencing and bounding during route search, there is the potential for a substantially increased cost in computation terms. A naive implementation, where each agent used route search to form a bid for each task, would incur computational costs on the order of the number of tasks multiplied by the total cost incurred in using the ATC approach. We expect that TA-IA approach will be somewhat more computationally expensive than the ATC approach. By expanding our bounding methods, however, the computational increase should be far less than a factor proportional to the number of tasks.[1]

The goal of an auction cycle remains the same as in the ATC approach - to allocate a single task to the task-addressing agent that can achieve the most in terms of average reward for addressing the task. Unlike the ATC approach, however, where bids were the results of a computationally inexpensive estimation of quality, bids in the TA-IA approach reflect full explicitly scheduled coordination solutions to reach and address the task in question.

4.1.1 TA-IA Implementation

There are two primary differences in terms of implementation between the TA-IA approach and the ATC approach: the structure of the auction and the bounding that occurs during route search. We first consider the structure of the auction, shown in Algorithm 4.

Algorithm 4: Structure of the task auction in the TA-IA approach

```
Input: T: List of unaddressed tasks
       A: List of fire truck agents
Result: Each idle fire truck is assigned a task, with corresponding bulldozer assignments
1  curBidMax ← 0 // Initialize so that any positive bid improves on it
2  T ← SortTasksByCurrentValue(T) // Sets task sequence for auction
3  foreach t in T do
4      foreach a in A do
5          bid ← BidOnTask(a,t,curBidMax)
6          if bid.reward > curBidMax then
7              curBidMax ← bid.reward
8          Add bid to bidList
9  winningBid ← DetermineHighestBid(bidList) // The highest bid will be
       associated with the final curBidMax
10 if winningBid is unset then
11     break // There were no good bids, so all agents must be occupied
12 AwardTaskToBiddingAgent(winningBid) // The winning agent will fix its
       schedule and task bulldozers
```

There are a few structural features of this algorithm which are designed to heuristically improve bounding. First, in line 2 the available tasks are sorted. We choose to sort them in order of current reward. Setting the order of the tasks will not change the outcome of the auction, but the order in which tasks are offered can affect the bounding that agents are able to use within the bidding process. We sort in order of current reward with the idea that high reward tasks will produce high value bids, which can then be used to bound future route searches as described later. Second, the maximum bid submitted for any task thus far is tracked in the *curMax* variable, and passed along with the task to the fire truck agents to use during bounding, as we describe next. Finally, we note

[1]This approach was originally presented in (Jones *et al.*, 2008)

that even if several agents are bidding on tasks, after a single task is awarded to some agent the bidding process must repeat for any remaining idle agents, as bulldozers may have commitments associated with the awarded task that must be reflected in future bids.

Algorithm 5: RouteSearchTA_IA - route search used in the TA-IA approach for a fire truck trying to determine the fastest route to a single target fire

```
Input: a: Agent that needs to route search
       f: Target fire
       C: List of currently assigned constraint-satisfaction duties and their scheduled
          completion times
       B: List of bulldozers
       m: Graph map of environment
       curBidMax: Current high bid for any task
Result: bestSched will contain a schedule and corresponding constraint allocationsthat
        improves on curBidMax in terms of average reward, or will remain unset if no
        route can improve on curBidMax
1  bestRew ← curBidMax
2  InitializeAstar(m, a, f, NUM_DEBRIS_CONS)
3  p ← GetNextShortestRoute()
4  s ← GenerateScheduleFromRoute(p,C)
5  r_shortest ← DetermineScheduleZeroDelayAverageReward(s)
6  if r_shortest < bestRew then
       // Even the shortest path assuming zero delay has inferior reward
7      return
8  for numCons ← 0 to NUM_DEBRIS_CONS do
9      InitializeAstar(m, a, f, numCons)
10     while true do
11         p ← GetNextShortestRoute()
12         if p is not set then
               // No more unique routes with this number of constraints.
13             break
14         s ← GenerateScheduleFromRoute(p,C)
15         r_zero ← DetermineScheduleZeroDelayAverageReward(s)
16         if r_zero < bestRew then
               // All remaining routes with this number of constraints cannot
                  improve on bestRew even assuming zero delay
17             break
18         if DetermineScheduleAverageReward(s) > bestRew then
19             s ← AuctionUnassignedConstraintsInRoute(a,s,B,bestRew)
20             if DetermineScheduleAverageReward(s) > bestRew then
21                 bestRew ← DetermineScheduleAverageReward(s)
22                 bestSched ← s
```

The route search process for TA-IA, shown in Algorithm 5, is similar to the route search ATC algorithm shown in Algorithm 3, with a few important differences. From the bulldozers' perspectives little changes - they still associate bids with particular auction contexts, though in this approach they will potentially be assigned constraint clearing duties associated with route searches associated with a number of different agents and tasks.

From the fire trucks' perspective the differences stem from the fact that route search is now framed in terms of reward rather than time, as the goal of the route search is to either determine a reward better than the current maximum bid for any task $curBidMax$ or to determine as quickly as possible that no such bid exists. In a given auction cycle even if several agents are bidding on tasks only a single bid associated with the single task schedule with the highest average reward will be awarded; the bounding algorithm can use this information to focus search exclusively on routes that have the potential to improve on the current maximum bid. In Line 1 we initialize *bestRew* to the current maximum bid, and will only consider schedules that can potentially improve on the best bid. The *bestRew* is also passed into the `AuctionUnassignedConstraintsInRoute` function, which is identical to the version presented in Algorithm 2 except that the time-based comparison is now an average reward-based comparison. If for a particular task no route is discovered that can improve on this best bid, then the *bestSched* variable will remain unset; the fire truck can then submit a NULL bid for the task.

Alterations to bounding can be seen in Lines 10 and 11. If a given route cannot improve on the current maximum average reward even assuming zero delay, then it must be the case that all remaining potential routes with greater length cannot improve on *bestRew*; remaining routes with that number of constraints need not be considered. Bounding can also be used to limit the paths considered to reach a fire to a single route. In Lines 2-8 of Algorithm 5 the shortest path with NUM_DEBRIS_CONS constraints or fewer is generated and tested for reward assuming zero delay. If even this shortest path does not improve on the current best bid then no other potential routes need to be considered.

We show an example of part of the operation of the TA-IA approach in Figures 4.1 and 4.2. In this example Fire X is the highest reward fire and is the fire upon which agents bid. The schedules determined during the route search process for fire trucks and bulldozers used to produce the bids as well as the average bids (in purple text) are shown in Figure 4.1. Note that bulldozer schedules may be provisionally committed to satisfy constraints associated with different fire truck's schedules. This does not result in conflicts in execution, as only a single provisional plan will be adopted during this auction cycle. The best bid among those submitted is Truck B's average reward bid of 22. The next fire in order of reward is fire W, and the $curBidMax$ of 22 is included in the auction call. In Figure 4.2 we illustrate Truck A's and Truck C's shortest path to fire W. In Truck C's case, even assuming no delay at constraint sites, it will take Truck C 12 cycles to address the fire if it takes the shortest path route assuming no delay. This will yield a reward of 236, and an average reward

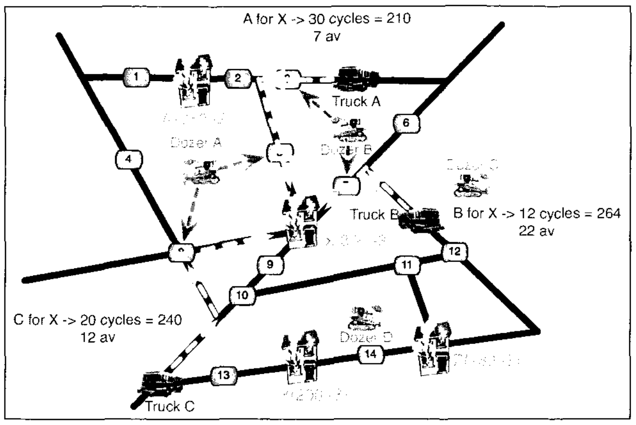

A for X -> 30 cycles = 210
7 av
Truck A
Dozer A
Dozer B
Truck B
B for X -> 12 cycles = 264
22 av
C for X -> 20 cycles = 240
12 av
Dozer D
Truck C

Figure 4.1: The schedules produced by agents in the TA-IA approach to determine bids for task X in the first auction cycle in a disaster response domain instance.

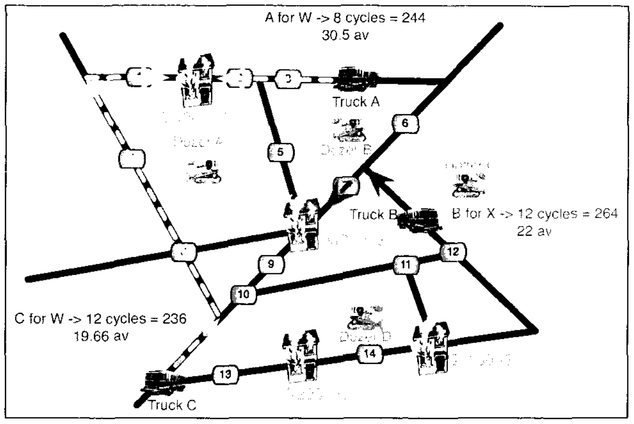

A for W -> 8 cycles = 244
30.5 av
Truck A
Truck B
B for X -> 12 cycles = 264
22 av
C for W -> 12 cycles = 236
19.66 av
Truck C

Figure 4.2: Truck C's and Truck A's shortest paths to fire W, the next fire proposed for bids in the auction cycle. Truck B's schedule, currently associated with the curBidMax, is also shown.

of 19.66. This value is less than $curBidMax$, and C need not hold sub-auctions or consider any other routes to the fire; it will submit a NULL bid for the task. In Fire A's case, taking its shortest path yields an average reward of 30.5 assuming no delay, which is greater than $curBidMax$. Truck A must do a more substantial route search process to determine if some explicitly scheduled route can improve on an average reward of 22.

Given that fire trucks are being assigned only a single task at a time, as they complete their tasks they will need to be assigned additional responsibilities. In some cases several trucks may be idle and competing against each other in an auction round. In other cases only a single truck may be idle, and will be the only bidder in an auction. The frequency of these cases will be the product of the auction frequency, the number of agents, and the average time it takes an agent to address its assigned task. In the latter case multiple auction rounds will need to be held, as each task assignment will generally have associated bulldozer assignments that will need to be accounted for in future auctions. In cases where there are multiple agents bidding on tasks, route searches can be executed in parallel, as our algorithm supports bulldozer bidding on auctions associated with separate contexts. The auctioneer can either wait until bids are received for a task before sending the auction call for the next task, including an up-to-date $curBidMax$, or can send the appropriate call to agents whenever bids are received.

4.2 Tiered auctions with time-extended task assignments

Tiered auctions allow the results of route search to factor into allocation decisions. Up to this point, however, all approaches have been limited in terms of time horizon as they do not search in the full space of task allocations and sequences. Allocating only a single task at a time is beneficial in that it limits the size of the search space; furthermore, we can exploit the fact that only a single task is assigned per auction cycle in bounding. In this section we present our approach to market-based multi-task allocation in domains with intra-path constraints. Considering the space of multi-task allocations and sequences adds exponential layers of complexity to the search space. One approach suggested by the literature for time-extended assignment would be a sequential single-item auction, a technique we described in Chapter 3.3. In independent coordination domains, agents can independently and efficiently assess many different task sequences for a particular allocation. In our domains of interest, however, determining the quality of a particular task sequence may involve considerable computation, as it requires searching the space of possible routes between tasks and determining constraint allocations and sequences. Furthermore, reasoning about altering fire truck schedules will require changing bulldozer allocation and sequences, which can affect the quality of all fire truck schedules. We could allow fire trucks to only consider adding tasks to the ends of their schedules, but that would produce solutions that are largely identical to the TA-IA approach. A naive implementation of sequential single-item auctions is unlikely to substantially improve in terms of solution quality on the TA-IA approach. [2]

We recognize that any approach that incorporates time-extended allocation and produces solutions quickly is going to need to heuristically focus search in the space of possible multi-task allocations. In our approach we leverage the spatial structure of domains with intra-path constraints to determine likely sets of tasks for joint allocation. One heuristic technique, clustering, identifies tasks that are near to each other and that to reach requires negotiating largely the same set of intra-path constraints. The bidding process for tasks then becomes a process of determining a maximum average schedule reward for addressing some or all tasks in a cluster. The second technique, opportunistic path planning, modifies the route searching process to reason about the potential for altering the route to reach other tasks on the way to a task cluster. The goal of route search in this view is to maximize the average reward over the length of a potentially multi-task route rather reaching a single task as quickly as possible. By employing both of these heuristic techniques we limit search to a small fraction of the potential multi-task allocations and sequences; this limits the increase in required computation compared to single task allocation approachs while improving the quality of the determined coordination solution.

In the rest of this section we describe these heuristics and how they are incorporated into the tiered auction framework. We additionally focus on altering bounding methods to accommodate multi-task assignment.

[2]This approach was originally presented in (Jones *et al.*, 2009)

4.2.1 TA-TE implementation

In this section we first discuss our clustering heuristic, describing how we adjust the auction process to hold cluster auctions instead of auctioning individual tasks. We then discuss the incorporation of opportunistic path planning during the route search. Finally, we talk about the function of the full system and how we use new bounding methods to limit the search space.

Clustering on shared intra-path constraints

Our approach to clustering is to group together tasks that are spatially close to each other and that are separated by at most a small number of debris piles; these tasks will be auctioned as a group. In our implementation clustering occurs at the auctioneer level. When initiating task auctions the D/A runs a clustering algorithm over all unassigned tasks. The algorithm iteratively assigns an unassigned fire task to a cluster and creates a cluster centered at that node. It then runs a **breadth-first search (BFS)** from the selected node. The BFS considers all unassigned fire tasks connected by edges with a total length less than a distance parameter CLUSTER_DISTANCE and that are separated by fewer debris pilesthan a parameter CLUSTER_DEBRIS_MAX; all such fire nodes that have not been assigned to another cluster are included in the new cluster. A clustering of a number of tasks is shown in Figure 4.3.

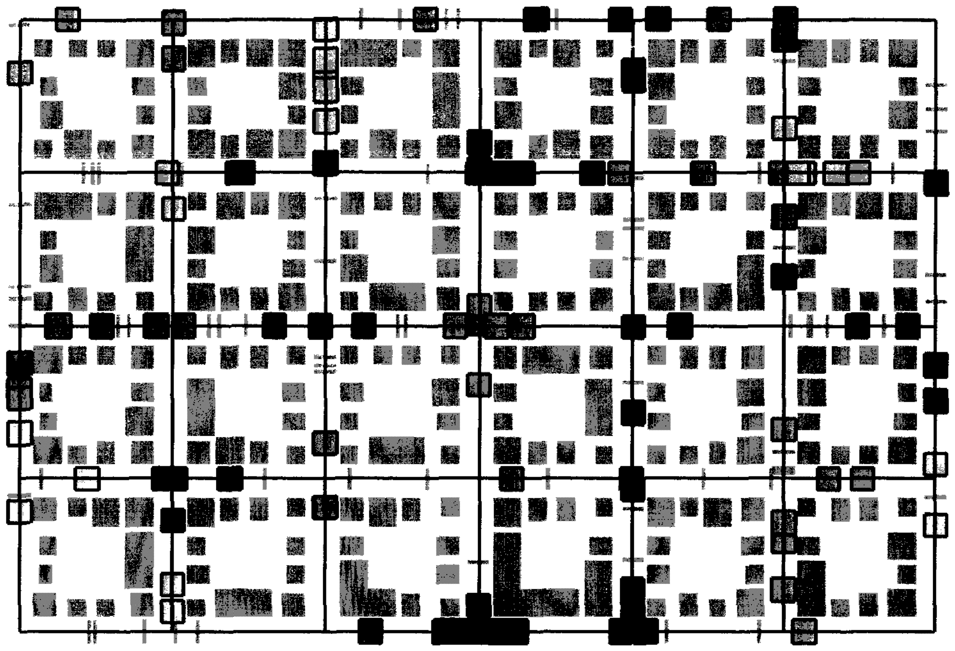


Figure 4.3: Output of the clustering algorithm with CLUSTER_DISTANCE = 2.5 and CLUSTER_DEBRIS_MAX = 5 on a disaster response scenario in a five-by-seven road network with 100 fire tasks (squares) and 100 debris (short lines that run perpendicular to the roads). Clusters are grouped by color.

Algorithm 6: `BidOnFireCluster` - used by fire trucks to bid on task clusters

```
   Input: a: Bidding agent
          c: Target cluster
          C: List of current commitments
          B: List of bulldozers
          m: Graph map of environment
   Output: bid: average reward for the determined cluster bid
1  currentLoc ← a
2  remainingTasks ← c
3  bidReward ← 0
4  while remainingTasks is not empty do
5      t ← DetermineClosestTask(remainingTasks,currentLoc)
       // RouteSearchTATE is described in Algorithm 7
6      candidateClusterSched ←
       RouteSearchTATE(a,t,bestClusterSched,bestClusterAwards,C)
7      if bidReward <DetermineScheduleAverageReward(candidateClusterSched) then
           // Candidate schedule incorporating t improves on previous schedule
8          bestClusterSched ← candidateClusterSched
9          bidReward ← DetermineScheduleAverageReward(candidateClusterSched)
10         bestClusterAwards ← DetermineFullAwardSet(candidateClusterSched)
11         currentLoc ← t
12     Remove t from remainingTasks
13 return bidReward
```

There are a few important differences between the tiered auction for task clusters and for single tasks. First, the D/A offers a cluster of tasks in each auction instead of a single task. Second, the fire trucks must now assess the reward that can be obtained from addressing a set of tasks rather than a single task. The algorithm fire truck agents use to determine cluster bids is shown in Algorithm 6. In order to determine a bid for the cluster, the fire truck must determine the order in which to address cluster tasks and the route to take between cluster tasks, as well as how to reach the first task in the cluster. Selecting the optimal ordering of tasks within a cluster so as to maximize reward is a difficult problem; we use a heuristic based on shortest path distance. The function call `DetermineClosestTask` takes a list of tasks and a location. It will determine the closest task by comparing the distance to each task along a shortest path whatever intra-path constraints lie along the selected path. Once a task has been selected, the task is passed into a route search algorithm tailored for tiered auctions with time-extended assignment - the algorithm for this version of route search is presented in Algorithm 7. The route search will take the existing schedule in *bestClusterSched*, if it exists, and add the results of route search from *currentLoc* to the task t, accounting for any awards associated with previous cluster segments in *bestClusterAwards*.

We expect that the initial route search, where the agent searches in the space of possible routes to reach the initially selected cluster location, to be the most computationally expensive, as the cluster may lie far away from the agent. Once one task in the cluster has been reached, however, route search should be easier, as the cluster tasks are near each other and separated by only a few constraint locations. Depending on parameterization some constraints may separate cluster tasks, so subsequent route searches after the initial search may require additional constraint assignments. In the auction call formation within the route search the context for the new cluster route search must be specifically linked to the previous cluster route search, as bulldozers that have previously be assigned debris piles associated with the cluster must account for those assignments in bids on later cluster constraints. The route search procedure for a fire produces a candidate cluster schedule, which is then compared to the best reward determined thus far in Line 7. The purpose for this comparison is to determine whether the extra reward gained by addressing the additional cluster task offsets the additional time required in terms of improving the average reward for the cluster schedule. The cluster bid need not incorporate all cluster tasks. If *candidateClusterSched* improves on *bestClusterSched* schedule, the candidate becomes the best schedule; *bidReward* and the necessary awards are updated, and *currentLoc* is set to the task location, as future plans for route search will need to begin from the end of the cluster schedule.

Once all cluster tasks have been considered, the value *bidReward* will be returned as the cluster bid. The auction algorithm will then award a cluster to the agent with the highest bid, assigning the winning agent all the tasks associated with the cluster bid. The task-addressing agent and associated constraint-addressing agents can then adopt the schedules associated with the cluster. If one or more task-addressing agents remains untasked the algorithm will re-do clustering to account for the new task assignments, and another auction cycle will be run.

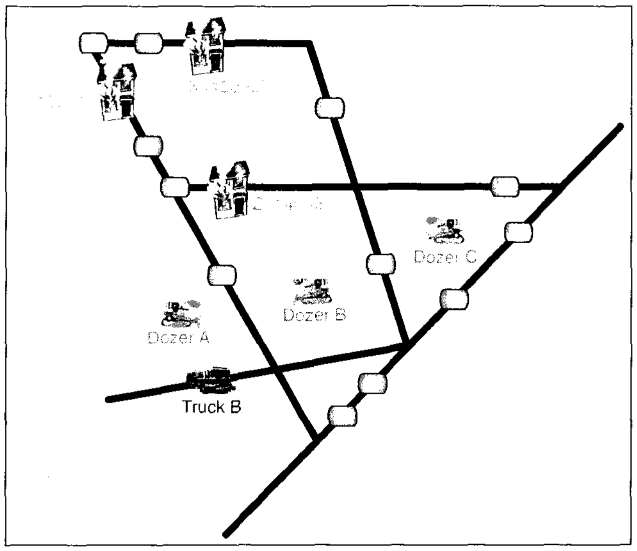


Figure 4.4: Truck B considers forming a bid for the cluster of tasks consisting of X, Y, and Z.

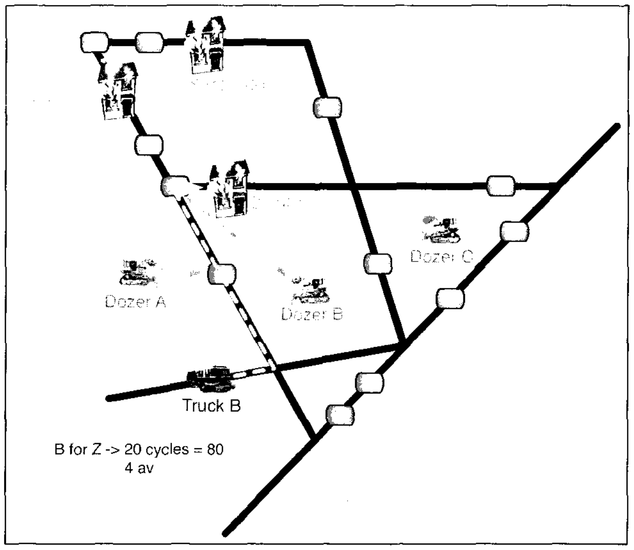
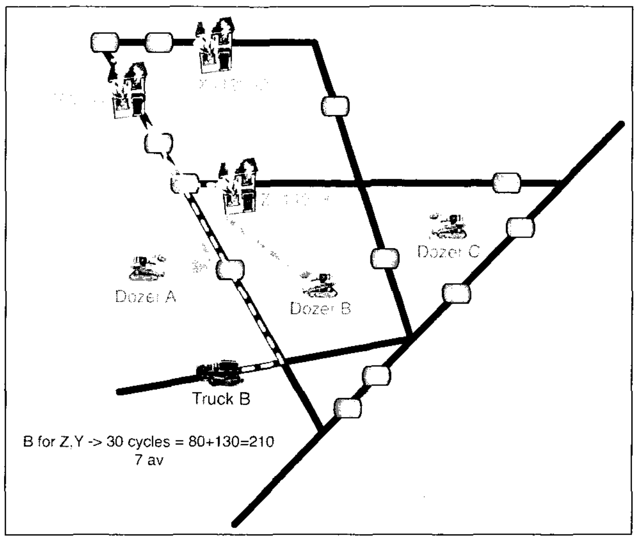


Figure 4.5: Truck B chooses fire Z as the first task to evaluate as it is the closest in terms of shortest path distance from its current location. The route search sub-auction procedure produces the route and bulldozer assignments associated with context 10111.

We illustrate the cluster auction process in Figures 4.4 through 4.7. Figure 4.4 shows a scenario where Truck B is bidding on a cluster of three fire tasks. The first step of the bidding process is to sort the fires by shortest path distance. In this instance fire Z is the closest to the agent and gets considered first. The tiered auction route planning process for Z yields the schedule shown in Figure 4.5, with an average reward of 4. The truck then sorts the remaining fires by the shortest path distance from Z, and selects fire Y as the next closest fire. The schedule determined by route search including both Z and Y is shown in Figure 4.6. The average reward for this schedule is 7, improving on the previous best of 4, and the new schedule is adopted as the cluster schedule. Finally, the truck plans a route from Y to X, shown in Figure 4.7, with bulldozers again bidding based on the combined context 10113. In this instance the fire truck determined that taking a longer route from Y to X that passed through only one debris was faster than waiting for two debris to be cleared on the shorter route. The average reward for this route is 4.6, which is lower than 7, so the schedule for addressing X in addition to Z and Y is discarded. The fire truck will bid 7 for the fire cluster representing the final average reward for the determined cluster schedule.



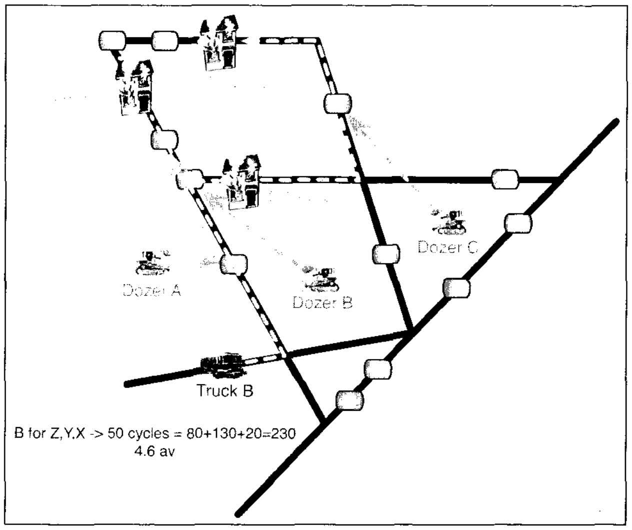


Figure 4.6: Truck B next plans from Z to Y, the closest remaining cluster task. It uses a sub-auction to determine the indicated route and debris assignments, explicitly linking context 10112 to context 10111 so that bulldozers bid based on the schedules associated with context 10111.

Figure 4.7: Truck B plans a route from Y to X. A bid for the cluster can now be calculated.

Opportunistic path planning

We next focus on our second heuristic approach to time-extended coordination - opportunistic path planning. In all of our tiered auction work described thus far the goal of route planning has been to determine which route will allow the fire truck to reach and extinguish a target fire as quickly as possible. In opportunistic path planning the goal of route search is instead to determine the route to the target fire that will allow the fire truck to accumulate the greatest average reward, where reward can come both from addressing the target fire and from attending to additional fires along or near the route. In many cases taking a somewhat more indirect route to a target fire may result in higher average reward. We use two different techniques to identify fires along particular routes that can be opportunistically added to the route. The first technique reasons about adding unallocated fires that lie directly along the node sequence that constitutes the route. The second technique reasons about adding tasks that require slight deviations from the route. When considering whether to opportunistically address a task we must determine whether or not the reward gained by addressing additional tasks along the route outweighs the reward lost by delaying the completion of tasks later in the route.

On-path tasks The on-path segment of our opportunistic path planning approach reasons only about tasks that lie directly along the route being evaluated. In this algorithm we start the evaluation at the first node in the route. If this node is an unassigned fire – a list of unaddressed, unassigned fires is passed into the algorithm – we create a new route schedule that is an exact copy of the current route schedule except that the new schedule will include a stop at the unassigned task for the required duration to address the task. The rewards for the schedules are then compared; whichever schedule has the higher per-cycle average reward is adopted and the search continues until the end of the route schedule.

Off-path tasks The off-path segment of our opportunistic path planning approach reasons about adding tasks that do not lie precisely along the given route but require only a slight deviation from the current route. We note that the primary points where a truck can beneficially deviate from the route lie at the beginning of the route and at each intersection. Thus at each of these points we run a distance-limited breadth-first search; the search will continue until the distance limit parameter OPP_MAX_DISTANCE is reached or debris are encountered. When the search finds an unassigned fire a new route schedule is created with the additional required detour to reach the candidate fire and return to the route. If this new route schedule achieves greater average reward we adopt it; otherwise it is discarded. The process continues until no additions can improve the route. In practice setting a high value for OPP_MAX_DISTANCE can improve performance - our use of average reward ensures that only tasks that improve performance will be added, and constraints generally serve to limit the space in practice.

Route search for time-extended coordination

Algorithm 7: `RouteSearchTA_TE` - route search used in the TA-TE approach for a fire truck trying to determine the highest average reward route to a single target fire given that the schedule may already include reaching previous fires.

```
Input: a: Agent that needs to route search
       f: Target fire
       curSched: Current cluster schedule
       curAwards: Awards already made in determining curSched
       C: List of currently assigned constraint-satisfaction duties and their scheduled
          completion times
       F: List of currently unaddressed, unassigned fires
       C: List of current commitments
       B: List of bulldozers
       m: Graph map of environment
Output: A schedule bestSched that consists of curSched followed by a route which achieves
        high average reward between the last node of curSched and f
1  bestRew ← 0
2  gvalMax ← HIGH_VAL
3  if curSched is set then
4      startLoc ← curSched.lastNode
5  else
6      startLoc ← a
7  for numCons ← 0 to NUM_DEBRIS_CONS do
8      InitializeAstar(m,startLoc,f,numCons)
9      while true do
10         p ← GetNextShortestRoute(gvalMax)
11         if p is not set then
               // No more unique routes with path cost less than gvalMax with this
                  number of constraints.
12             break
13         s ← AppendRouteToSchedule(p,curSched,curAwards,C)
14         PotentiallyAddOnPathNodes(s,curSched,F)
15         PotentiallyAddOffPathNodes(s,curSched,F)
16         if bestRew < DetermineScheduleAverageReward(s) then
17             s ← AuctionUnassignedConstraintsInRoute(a,s,B,curAwards)
18             if DetermineScheduleAverageReward(s) > bestRew then
19                 bestRew ← DetermineScheduleAverageReward(s)
20                 bestSched ← s
21             timeToComplete ← bestSched.compTime − curSched.compTime
22             if timeToComplete < gvalMax then
23                 gvalMax ← timeToComplete
```

The route search algorithm used in TA-TE is shown in Algorithm 7. The two opportunistic path planning algorithms for on- and off-path tasks are called in Lines 17 and 18. There are a few important differences between this algorithm and the one used by TA-IA. One set of changes involves accounting for the fact that the current route search may just be one segment of a larger plan to address a cluster. This primarily affects the process of transforming a route into a full schedule given the results of previous cluster route searches, and requires passing the relevant information into the auction function so that bulldozers will be informed of which context schedules to use in bidding.

The biggest differences are associated with bounding. In the previous route search algorithms, both for ATC and TA-IA, bounding was done on the basis of finding the fastest route to a fire. When considering opportunistic path planning the goal is instead to maximize the average reward for the schedule segment given that a slower route may offer additional opportunities to gain reward from addressing fires. Considering only routes that may allow for fast passage to the task location may ignore somewhat slower routes that would offer greater reward density. At the same time, allowing arbitrary levels of indirectness can potentially be harmful to performance. If every path is considered, no matter how indirect, then the search will be very slow as the full space of non-repeating paths will be considered in every route search. Additionally, in some cases indirect paths may require gaining passage through additional constraints, potentially taxing constraint-addressing resources.

The method we adopt for bounding is to allow only limited indirection. We achieve this through g-value bounding. To explain, consider line 26 in Algorithm 7: even if a schedule does not improve on the best schedule in terms of average reward, if the new schedule is the fastest plan to address the task the time taken to address the task is recorded in $gvalMax$. This value is then passed into the A* algorithm. The g value in the A* algorithm is the traditional variable for signifying the path cost of a given variable. We incorporate the g-value bound into the A* algorithm by instructing the algorithm to only return a path if the cost of the path is less than the current g-value bound represented by $gvalMax$; only paths that have a cost in terms of distance that is less than the current value of $gvalMax$ will be considered.

Note that as implemented the method for setting $gvalMax$ already has slack built-in, as it includes the entire duration associated with addressing the task, not just the path distance. Thus any delays associated with addressing additional tasks or waiting for constraints to be satisfied will lead to higher values for gvalMax. Our method of setting the maximum g-value is only one possible method for setting g-values. We could also associate the g-value with some factor of the shortest path distance, and limit the search to paths with a length no greater than that value.

The g-value bounding method combined with the NUM_DEBRIS_CONS parameter serve to prevent too much time being consumed searching increasingly indirect paths, and to prevent the over-exploitation of available constraint-addressing resources.

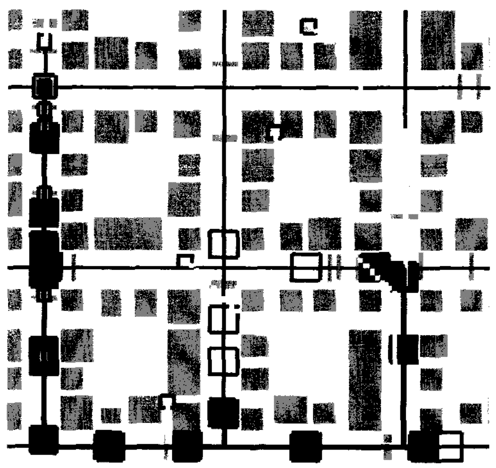
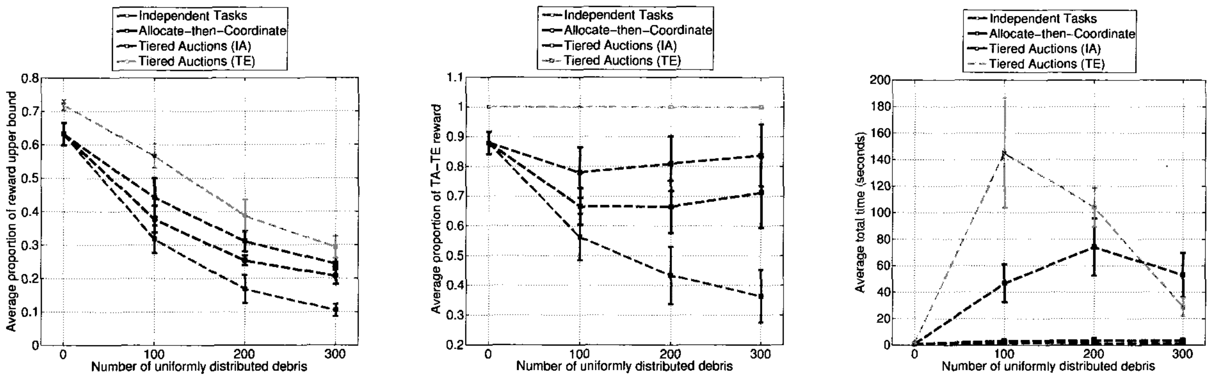


Figure 4.8: An example from our simulator visualizer of a route determined using clustering and opportunistic path planning. Red filled squares are fires that are scheduled to be addressed. Debris piles are represented by pink lines perpendicular to the road network. Small green boxes are bulldozers with their associated paths in yellow. The blue square within the orange square in the upper left is the fire truck agent.

The schedule for one agent that results from coordinating using TA-TE can be seen in Figure 4.8.

4.3 Experiments and evaluation

In this section we examine the performance of our tiered auction approaches versus the minimal time approaches described in Chapter 3. Our domain instances for testing are identical to those described in Chapter 3.6. We test with four approaches - the independent tasks and ATC approaches as described in the previous chapter, tiered auctions with instantaneous assignment (TA-IA), and tiered auctions with time-extended assignment (TA-TE). We set a value of 12 in each of the tiered auction approaches for NUM_DEBRIS_CONS; this was experimentally determined to represent a good balance in terms of improved solution quality and minimum computation time. Increasingly this value leads to a greater number of paths being considered, which slows search and can lead to greater overexploitation of bulldozer resources. If the parameter is set too low it can prevent agents from being able to determine any path to a number of fires, harming performance in domains with high constraint densities. In the tiered auction approach we set a value of 2.5 for CLUSTER_DISTANCE, 5 for CLUSTER_DEBRIS_LIMIT, and OPP_MAX_DISTANCE. These parameter settings were determined arbitrarily. Increasingly the values can lead to larger clusters, which are more computationally expensive to assess. We set the g-value multiplier to 1.0 - as described in Section 4.2.1 this still represents a somewhat loose bound. This was experimentally determined to be a reasonable setting for the bound.



(a) Average reward achieved by approaches as a proportion of an upper bound of available reward, where the bound is the summed reward of all issued tasks if they were addressed immediately. Standard deviations are shown as error bars.

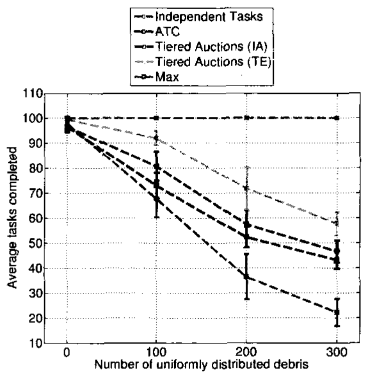
(b) Average reward achieved as a proportion of the reward achieved by the TA-TE approach on a per trial basis.

(c) Average total time taken to produce coordination solutions. Trials were all run in a single thread and allowed 100% of a CPU on a quad-core Intel Xeon 3.8 GHz CPU.

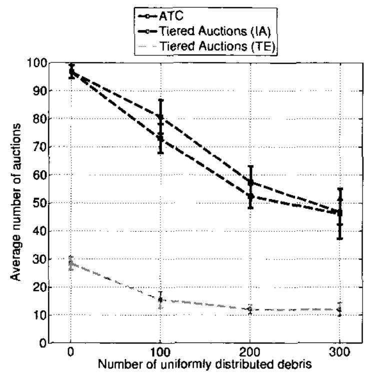
Figure 4.9: The performance of the tiered auction and minimum time approaches in a simulated disaster response domain

Figures 4.9(a) and 4.9(b) show the differences in solution quality between the two approaches. Figure 4.9(a) uses the same scale as Figure 3.10(a), but Figure 4.9(b) attempts to distinguish the comparison between approaches on individual trials rather than average received reward. Different domain instances can be dissimilar in terms of available reward or the ease with which that reward can be acquired. To this end, in Figure 4.9(b) we compare the percentages of reward obtained in individual trials versus that of our best performing approach, TA-TE. We first consider the performance of TA-IA versus the minimal time approaches. When considering the average performance for domains with constraints, TA-IA outperforms ATC at all levels, with an average margin of 19%. The performance margin between TA-IA and independent tasks is even greater. Using tiered auctions can improve performance substantially over the ATC approach. Looking at Figure 4.9(b), we can see that using time-extended allocation can substantially improve performance. TA-TE outperforms TA-IA at all levels, with an average improvement of 24% in domains with constraints. TA-TE outperforms the minimal time approaches by even more substantial margins.

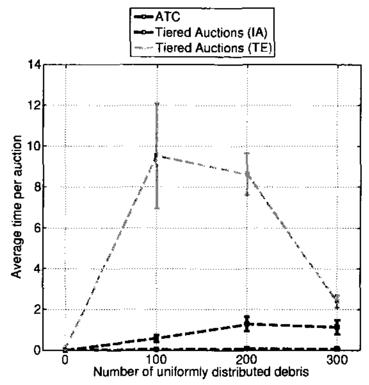
Total time taken in the different approaches is shown in Figure 4.9(c). Both of the tiered auction approaches are substantially more computationally expensive than the minimal time approaches. In the trials with constraints, TA-IA uses almost 18 times more computation than the ATC approach. While this is a substantial increase in cost, it is far less than would be incurred in a naive approach. TA-TE uses substantially more computation than TA-IA at the 100 and 200 constraint levels, but slightly less at the 300 constraint level. Overall it uses 60% more computation in domains with constraints.



(a) Average number of fire tasks out of the 100 issued that were addressed before reward values reach zero



(b) Average number of auctions conducted during a single trial



(c) Average time taken per auction

Figure 4.10: Other aspects of approach performance in a simulated disaster response domain

In addition to solution quality and total computation time there are a number of other distinctions between the approaches. Figure 4.10(a) shows the number of tasks addressed before task reward reached zero. Each of the approaches address almost all tasks when there are no constraints, but the number of tasks addressed decreases precipitously for all approaches as intra-path constraint density increases. TA-TE successfully address the greatest number of tasks. The differences between the instantaneous and the time-extended approaches become more pronounced when we examine Figure 4.10(b), which shows the number of auction cycles held by the explicit scheduling approaches. For the ATC and TA-IA approaches this mirrors the number of tasks addressed, as an auction cycle is held for each task assignment. For the TA-TE approach, however, a number of tasks can be allocated in each auction can be substantially greater than one; this is reflected in the greatly decreased number of auctions held. Figure 4.10(c) shows the average computation time per auction cycle. In all domains with constraints the TA-TE approach takes more time per auction cycle, taking more than 20 times as long at the 100 constraint level. The time per auction for the TA-TE approach decreases dramatically at the 300 constraint level - at high levels of intra-path constraint density the NUM_DEBRIS_CONS becomes an increasingly strong bounding tool, leading to the decrease in computation time.

These results are especially interesting within the context of potentially interleaving planning and execution. The TA-IA approach holds auctions more frequently, but the auctions last for only a short time; the TA-TE approach holds auctions less frequently but they take a good deal more computation in many domains. The particular features of a domain scenario can affect which of

Approach	High constraint density	Little time for planning	Communication cost
Ind. Tasks	−−	++	++
ATC	−	+	+
TA-IA			−
TA-TE		−	−

Table 4.1: General guidelines on the strengths and weaknesses of the approaches in static domain instances with different characteristics. Strengths are denoted with the + symbol, and weaknesses with the − symbol, with particular strengths or weaknesses doubling these labels.

these computation profiles is superior.

4.4 Summary and discussion

In this chapter we introduced the tiered auction mechanism, which supports using sub-auctions in order to solicit the involvement of other agents during the bid formation process. This mechanism permits market-based approaches that can perform route search with provisional constraint allocation and scheduling as part of the bidding process. Determining explicitly scheduled coordination during task allocation substantially improves the quality of the allocation decisions. This improvement comes with an associated increase in computational cost, though we attempt to minimize the increased cost by bounding during the task bidding process in addition to bounding during route search.

We presented two algorithms, both of which use the tiered auction formulation. The first algorithm, tiered auctions with instantaneous assignment, considers only single task allocations. The second algorithm, tiered auctions with time-extended assignment, potentially considers multi-task allocations. Rather than considering the full, exponential space of multi-task allocations, the TA-TE approach heuristically groups tasks using two methods. The first - clustering - groups tasks that are geographically near each and separated by only a few constraints for joint consideration during the task auction process. The second technique - opportunistic path planning - reasons about the potential to address other tasks during the route search process. Both the tiered auction approaches substantially outperform the minimal time approaches in terms of solution quality; the time-extended tiered auction approach substantially improves on the instantaneous tiered auction approach. In terms of computation time both tiered auction approaches require substantially more computation than either of the minimal time approaches. In most domain settings TA-TE requires more computation than TA-IA, but the increased computation requirement is not prohibitive from a tractability perspective.

Table 4.1 summarizes the relative strengths and weaknesses of the four approaches introduced thus far in static domain instances. The two tiered auction approaches are the best performing of our market-based approaches, especially in domains with high constraint density; in these domains considering intra-path constraints during task allocation can help to limit delay at constraint sites, leading to improved performance. They both require substantially more computation time than either of the minimal time approaches, but improve considerably on the performance of those approaches. For scenarios where there is little computation time available, the computation requirements for the tiered auction may be prohibitive. Furthermore, both the independent tasks and the ATC approaches are minimal in terms of communication requirements, while the tiered auction approaches will potentially use a good deal of communication due to running a number of different route searches in each auction cycle. Thus for domains where communication is limited the minimal time approaches may be more appropriate. However, we believe that our tiered auction approaches, particularly the TA-TE approach, offer an excellent balance of solution quality and computation time. We consider it unlikely that an approach that uses substantially less computation time could rival our methods in terms of solution quality.

These approaches are still heuristic, however. Agents make greedy choices based on their individual rewards rather than reasoning about global solution quality; of particular concern is the requirement for parameter tuning to prevent the over-exploitation of constraint-addressing resources. Furthermore, using the earliest completion time heuristic rather than a more extensive search of the space of possible constraint allocations and sequences is essential for minimizing computation time, but means that many potentially beneficial constraint assignments and sequences are not considered. Finally, even the time-extended approach reasons about a small fraction of the potential multi-task allocations, exploring only a fraction of the full coordination space in static domain instances. The question remains what we give up in terms of solution quality by employing these heuristics.

Chapter 5 Full coordination space search approaches

In Chapters 3 and 4 we explored techniques that depend on heuristics, short planning horizons, and partial independence assumptions to limit the search space can determine high quality solutions quickly for domains with intra-path constraints. Employing this array of assumptions to narrow the search space means that even in the TA-TE approach the full coordination space is not considered. There are four primary sources of potential inefficiency inherent in the TA-TE approach, the best performing of our market-based approach in terms of solution quality. First, allocation is greedy – tasks are awarded on the basis of which agent can gain the most individually instead of how responsibilities can be divided among agents to maximize the overall solution quality. Second, during route search agents are allowed unfettered access to intra-path constraint satisfying agents, meaning that a single fire truck can potentially tie up the bulldozer resources for long durations. While this can be ameliorated somewhat by using g-value bounding and using the NUM_DEBRIS_CONS parameter, as discussed in Chapter 4, but inefficient resource allocations can still result. Third, bulldozers reason only about adding constraint satisfaction duties to the ends of their schedules during sub-auctions, but not how re-ordering constraints could potentially benefit the overall solution. Finally, our approach to tiered auctions with time-extended assignment will not generally result in indefinite time-horizon allocations - the heuristics are not designed to allocate all tasks or to explore all task sequences for either fire trucks or bulldozers.

To address these limitations we designed a genetic algorithm (GA) approach for search in the full coordination space, with an indefinite time horizon for planning. One central design goal in this work was to engineer an approach capable of determining solutions that take into account the interactions of agents in order to determine globally high quality solutions, rather than using a more myopically greedy market-based approach. The approach should distribute intra-path constraint satisfaction resources in a manner that can potentially benefit all agents, not just the agent that wins an auction. Finally, we wanted to make the approach relatively non-heuristic. While heuristics can reduce the time required for search they can also result in finding local minima in the coordination space, which we would like to avoid. Making such an approach distributed was not a central concern, as considering the interactions of all different agents' schedules will be difficult to distribute. [1]

The genetic algorithm approach proved superior over some other full coordination space solutions we considered, including a simulated annealing approach. The GA offered several advantages. First, we could leverage techniques from other work done in related domains, specifically from the vehicle routing literature, which involves elements of path planning (Russell & Lamont, 2005) and from the job-shop scheduling literature, which involves a variety of different time-oriented constraints (Mesghouni *et al.*, 2004). However, none of this literature confronts a domain of the complexity of domains with intra-path constraints. Second, the GA approach is multi-hypothesis, unlike in simulated annealing, where a single solution is adapted over time. The multi-hypothesis approach seemed far more capable of searching in the full coordination space – a number of potential solutions can be evolved over time rather than focusing on a single narrow solution.

A substantial challenge in designing our GA approach are accurately accounting for the interactions between agent schedules. To ensure that all interactions are represented each genome needs to represent an entire solution to the coordination problem - a full set of multi-agent plans. At the same time, GA operators such as crossover and mutation will generally only effect a portion of the coordination space - a single agent's plan in many cases. Relatively small changes in a single agent's sequence can have profound implications on the overall solution quality though - if the change means that a fire truck takes one route to a task instead of another, for instance, it can mean that bulldozer plans that produced high quality outcomes now produce low quality outcomes. While ideally the GA approach would be non-heuristic to avoid the local minima that heuristics can introduce, to avoid heuristics altogether may make it difficult to find good solutions no matter how long search is allowed to continue.

This approach style is appropriate only for scenarios with a particular set of requirements. First, there must be a requirement that solution quality be maximized whatever the computational cost, as the genetic algorithm will take a good deal of time to determine high quality solutions. Searching in the full coordination space is likely to be computationally intensive, and the genetic algorithm approach will not make extensive use of heuristics to speed search in the hope that local maxima can be avoided. Second, it must be possible to centralize all necessary data to allow the algorithm to function in a centralized fashion. Third, there must be reasonable fidelity between simulation and execution in order to accurately evaluate solutions. To determine the quality of a particular coordination solution requires simulating the effects of agent actions to account for all interactions. This makes the fidelity of the world model to the real world important, as the quality of the solutions determined will be limited by how well the simulation maps to the world. The GA approach will have limited applicability in domains where levels of uncertainty are high - in

[1]This approach was originally presented in (Jones *et al.*, 2009)

such domains it will generally be counterproductive to run a computationally expensive full time horizon approach. We cover the final issue in more detail in the next chapter.

In the next section we discuss the specifics of our genetic algorithm implementation. Then, with our full suite of approaches introduced we will present an extensive evaluation of how the approaches function in different parametrizations for static instances of the disaster response domain.

5.1 GA implementation

The central question associated with designing our genetic algorithm approach is how to represent coordination solutions within our algorithm; this decision impacts the design of all aspects of the system. Making this decision difficult was the necessity of representing fire trucks routes in the solution. Genetic algorithms are infrequently used for path-planning applications, and we could not find examples in the literature of approaches for domains that include both allocation and path planning in problem encodings. One choice of representation for fire trucks would be to mirror the representation we used for the coordination space in Chapter 1.2.2. This encoding explicitly represents fire task allocations and sequences as well as the routes taken between them. This would be a reasonable approach, but we decided to take a different approach with the hopes of generating a more compact representation.

We noted during the development of the opportunistic path planning algorithm described in Chapter 4.2 that routes could be selected on the basis of their ability to allow a number of different tasks to be reached. Rather than representing the task allocation and sequences explicitly, we can instead represent the complete route each truck will take through the domain environment as well as whether or not the trucks will attempt to stop and address a fire that they pass. This representation does not explicitly encode a task allocation of fires to trucks only the intention to address fires; which agents which actually address tasks when two agents share an intention to address the task is determined during simulation for evaluation. This representation has the advantage that it allow for the exploitation of spatial structure while still being sufficient to fully represent the possible space of coordination solutions. In terms of spatial structure, traveling long distances through a domain with intra-path constraints is likely to be difficult and time-consuming. Thus while allocations and sequences that call for moving long distances through the environment are part of the search space, they are unlikely to be part of a good solution. It is far more likely that agents should attempt to address other nearby tasks. In this view, where an agent goes is more important than the assignment or order in which it addresses tasks. At the same time, the representation is sufficient to represent any given task allocation, sequence, and route between tasks.

Our GA approach consists of the following algorithm, invoked at time zero:

Algorithm 8: GeneticAlgorithm(S, A) - an approach for determining a coordination solution using randomized evolutionary search in the full coordination space.

```
Input: S: Current world state
       A: List of agents
Result: Schedules are set for each agent
1  genomePool ← InitializeGenomePool(NUM_GENOMES_IN_POOL)
2  for i = 1 to NUM_GA_ITERATIONS do
3      foreach g in genomePool do
4          g.fitness ← EvaluateGenome(S,g)
5          if g.fitness < bestGenome.fitness then
               // The genome improves on the best genome evaluated thus far
6              bestGenome ← g
7      hallOfFame ← UpdateHallOfFame(genomePool, hallOfFame)
8      genomePool ← SelectFromGenomePool(genomePool, hallOfFame)
9      genomePool ← DoCrossover(genomePool)
10     genomePool ← DoMutation(genomePool)
11 SetAgentSchedulesFromGenome(A,bestGenome)
```

We will discuss each aspect of the genetic algorithm design in detail, but there are a few important aspects of the overall algorithm. First, we do optimization for a fixed number of generations rather than attempting to determine when the solution has converged. Secondly, we track the best genome over all generations, not necessarily the best genome from the final generation. This genome is used to set agents schedules, which can then be executed.

Next we go into each of the central aspects of this algorithm in greater detail, starting with our method for encoding genomes.

5.1.1 Encoding

As discussed above, the choice of representation and the methods for encoding that representation are central to genetic algorithm design. The genome encoding must represent a complete solution for a given domain instance. For the fire truck agents we must represent both their full routes as well as whether or not they will attempt to address fires they pass. We represent full routes as an ordered list of nodes that the truck will attempt to reach. Each truck will have a node sequence in a given genome. This route will not represent a schedule - only by simulation during the evaluation phase can we determine how agent sequences will interact to produce schedules. A Boolean work vector encodes whether or not fire trucks will attempt to address fires they pass, with an entry for each fire node in a node sequence. If the entry in the work vector is 1 for a given fire node the truck will attempt to address the fire. The bulldozer solution encoding is more straightforward, as bulldozer paths need not be represented in the encoding. The bulldozer encoding consists of a sequence of debris piles to address. Thus our full genome representation consists of a node sequence for each fire truck, a Boolean work vector for each fire truck of length equal to the number of fire nodes in the fire truck's node sequence, and a debris sequence for each bulldozer.

5.1.2 Initialization

We use a randomized method to initialize genomes both to seed the first generation in the `InitializeGenomePoo` function and to introduce new individuals into the population in subsequent generations to maintain diversity. In this method for each fire truck we select a random fire node and determine a shortest distance path to that fire node. As there may be a number of shortest paths of equal length, especially if the road network is a grid, we introduce random jitter on edge lengths so that the algorithm will randomly select one of the set of equal length shortest paths. That shortest path node sequence becomes each fire truck's initial entry in the genome. Work vectors are initialized such that trucks will attempt to address all fires on the route.

Our approach to bulldozer initialization is more complicated. Debris piles should be uniquely assigned, as duplicate effort is wasted. Our initial implementation randomly divided all debris piles among the bulldozers, but this results in solutions where bulldozers may spend a good deal of time traveling among debris pile locations, which is generally inefficient. While even poorly initialized solutions can improve over generations, having higher quality initial solutions can speed the search process. The approach we used to reduce travel times is to do bulldozer assignments using a simple greedy method that functions much like a market-based approach. In this method we iteratively determine the earliest completion time among all unassigned debris piles for each bulldozer given that they must complete their other assigned tasks first. The lowest value among these, representing the earliest completion time for any debris pile given current assignments, is assigned to the associated bulldozer. Using this assignment strategy instead of a random strategy can substantially reduce the time each agent spends traveling as well as keeping the number of debris piles assigned to each bulldozer approximately equal. This method also ensures that each debris pile is assigned to only a single bulldozer and that all debris are assigned to some bulldozer.

5.1.3 Fitness evaluation

Another essential element of the genetic algorithm is determining the fitness of individual genomes - whether or not they represent good solutions to the domain. For our domains the function we are trying to maximize is the objective function for the domain - this becomes the fitness value. This value must be assigned to each genome in the pool so that selection can preferentially propagate high fitness individuals. The important factor in terms of the objective function is the time at which fires will be extinguished. Given our representation, this can only be determined by simulating the actions of agents in the domain, as we must consider how agents' routes and sequences interact.

We use event-driven simulation in our evaluation. We first note that we can determine the times at which bulldozers will clear debris without considering the fire trucks' actions in any way; thus we first simulate the bulldozers' actions. We assume that each bulldozer remains at the site of debris pile until the required duration to clear the debris has been satisfied and then moves to their next assigned debris pile. Given these assumptions we only need to compute the travel times between different debris nodes – we accomplish this using an optimal path planner on an occupancy grid of the environment – and add debris task durations for each debris in each bulldozer's sequence to determine scheduled clearing times for each debris.

Once all debris clearing times have been established given assigned bulldozer sequences we move on to the fire trucks. We assume that a fire truck moves between the nodes in its sequence without pause except when addressing a fire or when waiting for a debris node to be cleared; if a fire truck arrives at a fire that has already been addressed or is in the process of being addressed by another fire truck we assume that fire truck simply continues on to the next node. We keep a vector of each fire truck's scheduled arrival at the next node in its schedule. In each iteration of the simulation we determine which fire truck has the earliest arrival time at its next scheduled node. We then need to determine at what time the earliest arriving agent will depart from that node location. If the node is an intersection, an already addressed fire, or a fire for which the corresponding entry in the work vector is false then the agent will depart the node immediately, making the departure time the same as the arrival time. If the node is a debris node then the truck can depart immediately if the debris is scheduled to be cleared by the time the truck will arrive at the node; otherwise the truck cannot depart until the time at which the debris is scheduled to be cleared. If the node is a fire node that has not been addressed and for which the work vector entry is true then the truck will depart after the required duration to address the task has passed. In this case we add the scheduled reward for completing the task at the indicated time to a running fitness total. Once a departure time has been determined for the node we can set the truck's arrival time at its next scheduled node by summing the departure time with the travel time between the current and next nodes. We then continue the process with whatever agent has the earliest scheduled arrival at the next node. We halt the simulation when all trucks have exhausted their schedules or when the earliest arrival time is past some horizon cutoff. The final fitness evaluation for the genome is the total summed reward gained by fire truck agents.

5.1.4 Selection

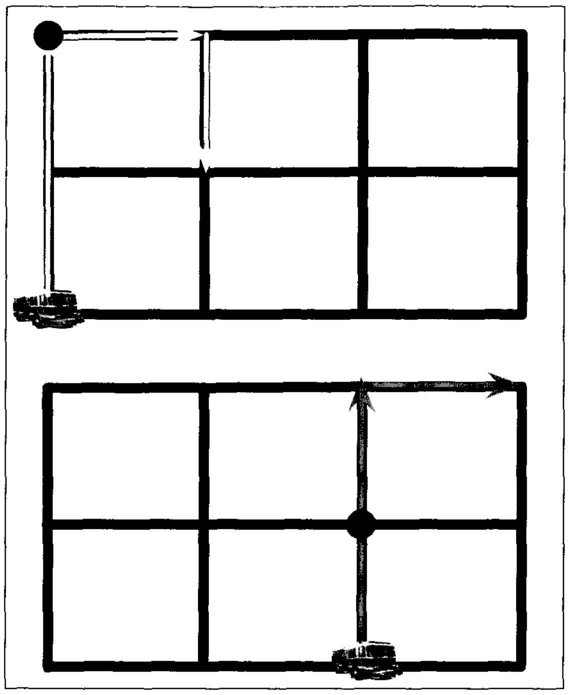
The goal of selection is to take the members of one generation of the genome pool and create a reproduction pool of individuals for populating the next generation. The reproduction pool will generally have higher average fitness than the genome pool, as the selection methods are designed to preferentially select higher fitness individuals. We use three methods to select individuals to populate this reproduction pool. The first method we use is a binary tournament method. In this method two members of the genome are randomly selected, and the member with the higher fitness is added to the reproduction pool. In this method higher fitness individuals are generally selected but a tournament may involve two low fitness individuals, maintaining diversity. The second method is **hall of fame (HOF)** selection. We maintain a list of a set size - P - of the distinct genomes with the highest fitness that have existed in any generation during the search - this is the Hall of fame. During the update process in Algorithm 8 we combine the current HOF and all genomes from the current pool and sort the entire list by fitness. The top P unique individual genomes are the HOF for the next generation. During HOF selection we pick a random individual from the hall and put it in the reproduction pool. Using the HOF method for selection insures that we spend a substantial portion of search time is spent exploring promising hypotheses. Finally, we also can add a genome randomly initialized using the initialization method described in Section 5.1.2 to the reproduction pool. Adding random genomes can help preserve genome diversity.

The average percentage of the time that binary tournament selection, HOF, and random selection are governed by the parameters SELECT_TOURNAMENT, SELECT_HALL_OF_FAME, and SELECT_NEW_GENOME.

5.1.5 Crossover

Once a reproduction pool has produced using selection we randomly alter pool members using two methods - crossover and mutation. These are the methods that create diversity among the genome pool and enable the genetic algorithm to explore the coordination space. The idea of crossover is to take two individual genomes and exchange parts of their solutions, with the hope that transferring segments from two high quality solutions will lead to even higher quality individuals.

Crossover can potentially occur for each adjacent pair of genomes in the reproduction pool. The probability that crossover will occur for each genome pair is determined by a parameter CROSSOVER_PROB. We use two different crossover methods - one for fire truck node sequences and another for bulldozer sequences. If a pair is selected for crossover both fire truck crossover and bulldozer crossover will occur. For fire truck crossover we randomly select a parametrized number – NUM_FIRETRUCK_TO_CROSSOVER – of fire truck sequence pairs from the genomes, where a pair consists of one sequence from each genome. We then use a mechanism that exchanges a part of each sequence. While work in genetic algorithms for vehicle routing often use somewhat heuristic approaches to insert sub-routes in beneficial points in agents' schedule (Russell & Lamont, 2005), we chose to use a simple non-heuristic approach, leaving other approaches for future work. The challenge in crossover is to maintain schedule correctness - each newly created sequence needs to be an executable route through the environment. We illustrate the process we use in Figures 5.1(a) and 5.1(b). We first divide each sequence into a front portion and a back portion by randomly selecting a position in each schedule as a dividing point – these points are shown as circles in the figures. The crossover method is then to make two new schedules consisting of the front portions of each sequence and the back portions of the other sequence. These may not represent executable schedules however as the last node in the front portion of each sequence may not be adjacent to


(a) Example of fire truck routes before crossover. Fire trucks node sequences from different genomes are shown in the top (light blue arrows) and bottom (magenta arrows). Points in the sequence selected for crossover points shown in top (dark blue) and bottom (dark red).

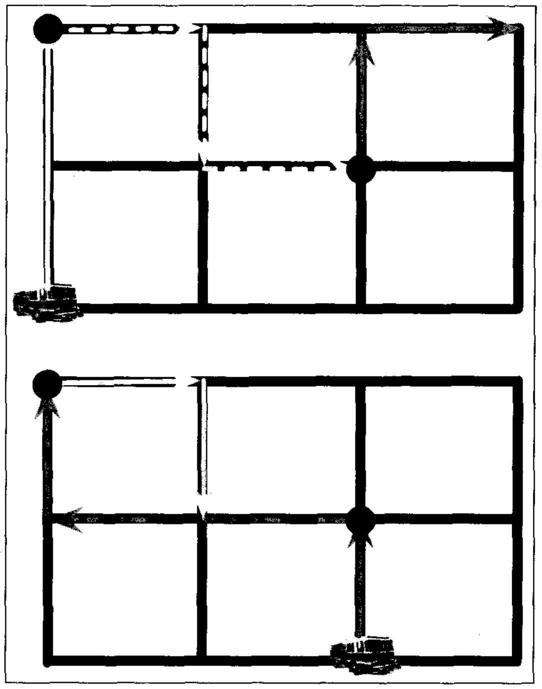

(b) Results of the crossover. Dotted lines represent generated shortest path segments between crossover points.

Figure 5.1: An example of fire truck crossover.

the first node in the opposite back portion. Thus we connect these two potentially non-adjacent nodes in each new sequence with a shortest path segment, produced using the random jitter method described in Section 5.1.2 – the shortest path sequences connecting the two crossover point nodes are shown as dotted lines in the figures. The result are two valid routes through the environment. Relevant portions of work matrices are carried over to the new routes.

Bulldozer crossover has the additional complicating factor that we want to maintain the uniqueness of assignments in each genome. To this end we used the crossover algorithm discussed in Mesghouni et al. in their work on genetic algorithms for job-shop scheduling (Mesghouni *et al.*, 2004). The basic idea of the approach is to select two bulldozer sequences from two different genomes and swap them. The complication comes from the fact that swapping sequences in this fashion could lead to duplicate constraint assignments and some constraints going unassigned, both of which we want to avoid. The approach we take is to provisionally make the assignments, and then to do two separate determinations. The first determination is whether or not the new sequences duplicate constraints. If so the entries for the repeated constraints are deleted from the crossed-over sequence.

Second, a list is compiled of any unsatisfied constraints that are now not assigned to any agent. These constraints are added to the end of the crossed-over schedule.

5.1.6 Mutation

Unlike crossover, mutation operators act on a single genome in isolation. The goal of mutation is to introduce small, random alterations to genomes. For the most part, our mutation operators are non-heuristic, in that they make alterations to agent sequences without reasoning about whether or not the alteration will improve performance. We also use one heuristic mutation operator on bulldozer schedules so that bulldozer tend to address the debris piles that fire trucks will reach first.

There are three parameters that govern the frequency of mutation - FT_NODE_DOANYMUTATION governs the average percentage of genomes in an iteration for which any fire truck sequence mutation is done, FT_WORK_DOANYMUTATION defines the average percentage of genomes that will be subject to work mutation, and BD_DOANYMUTATION specifies the average percentage of genomes that will be altered by bulldozer non-heuristic mutation.

Fire truck mutation operators

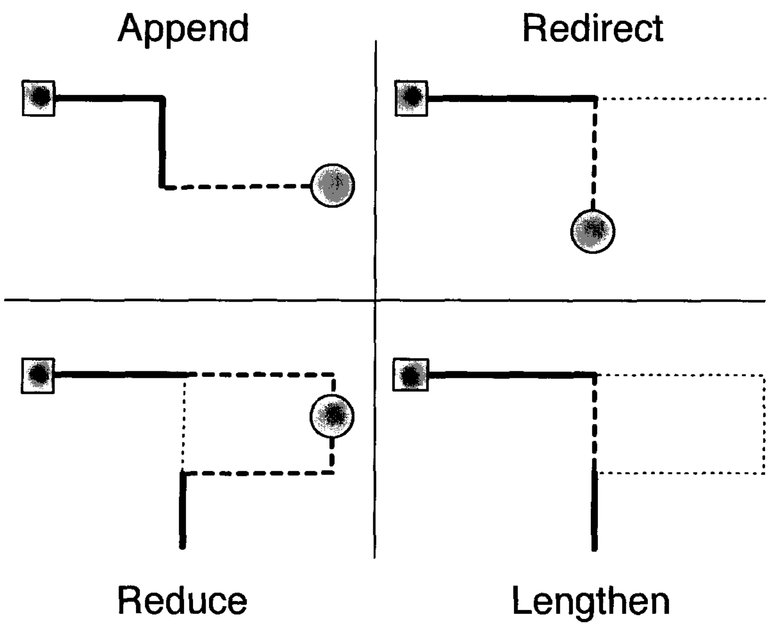


Figure 5.2: Examples of the fire truck mutations. The square is the agent's location, the solid line the unaffected part of the original sequence, the smaller dashed line the replaced sub-sequence, the larger dashed line the new portion of the sequence, and the circle a randomly selected target node.

The genetic algorithm needs to have the capability to explore the space of possible agent routes, and we needed mutation operators capable of altering routes to search this space. We could not find examples in the literature of mutator operators applied directly to agent routes. Our primary goal for the mutation operators was to have sufficient richness to take any route and mutate it into any other route in some finite number of steps. Each operator directly alters the path of a single fire truck. The first operator, "Append", adds to an agent's sequence by selecting a random target node, determining a shortest path from the final node in the agent's schedule to the target node, and appending that sequence to the existing sequence. The next operator, "Reduce", first selects two random nodes in the agent's sequence - the sequence between these two selected nodes represents a sub-sequence. The algorithm then determines a shortest path between the two nodes and substitutes the shortest path for the existing sub-sequence. The third operator, "Lengthen", also selects two nodes in an agent's sequence. It then selects a target node in the environment, and plans a shortest path from the node that appears earlier in the agent's sequence to the target node, and then another shortest path from the target node to the second selected node. These paths are then substituted for the existing sub-sequence. The final operator, "Redirect", selects a single node in an agent's sequence and a target node, planning a shortest path from the selected node to the target node. This shortest path is substituted for the part of the agent's schedule that follows the selected node. These operations are illustrated in Figure 5.2. Work vectors are set to 1 for each newly created segment. What percentage of the time a given fire truck mutation will be of each of the four forms is given by parameters FT_MUTATION_APPEND, FT_MUTATION_REDUCE, FT_MUTATION_LENGTHEN, and FT_MUTATION_REDIRECT.

Work vector mutation is simple - for each work vector entry there is probability FT_WORK_-SWAP_PER that an entry will be considered for alteration. Once considered, if 0 there is probability FT_WORK_ON_PER that it will be switched to 1, and if 1 there is probability FT_WORK_OFF_-PER that it will switched to 0.

Bulldozer mutation operators

We chose to use three standard mutation operators from the job-shop scheduling literature (Mesghouni *et al.*, 2004): swapping task order in a single agent's schedule (intra-swap), swapping two tasks between two different agents' schedules (inter-swap), and reassigning a task from one agent to another (assign). All mutation operations preserve the unique assignment of debris. The average percentages that a given non-heuristic bulldozer mutation will be of a given kind are governed by parameters BD_MUTATION_INTRA_SWAP, BD_MUTATION_INTER_SWAP, and BD_MUTATION_ASSIGNED.

During the development of the genetic algorithm we noticed that performance was relatively poor in domains with high densities of intra-path constraints. This poor performance is likely the product of a number of factors - in domains with high constraint density, even the optimal solution may not improve substantially on the heuristic solutions, or local maxima may be especially hard to escape. We were concerned, however, that in instances with many constraints having the GA employ only non-heuristic operators meant that the majority of search time was spent exploring poor solutions. One common source of inefficiency was in bulldozer assignments - even though the market-based assignment produces reasonable solutions, there is no guarantee that the genetic algorithm was going to produce particularly well-coordinated behavior. Bulldozers may spend considerable time clearing debris piles that are not on any agents route, or will clear debris out of the order that trucks require it.

In order to rectify this inefficiency we implemented a "smart" swapping heuristic mutation method that is intended to produce better coordinated schedules. The basic approach we take is that bulldozers will re-organize their sequences such that the debris piles that fire trucks will reach first are cleared first. Each genome will be subject to some number BD_NUM_SMARTSWAPS each iteration. A required piece of data in this approach is to determine the order in which fire trucks will arrive at debris piles. This is determined in the evaluation function. Each genome has an associated debris arrival ranking list. During the simulation as fire trucks arrive at debris piles, if those piles are not already in the list they are added to the end of the list. The result is a sequential ranking of the order in which fire trucks will reach debris piles.

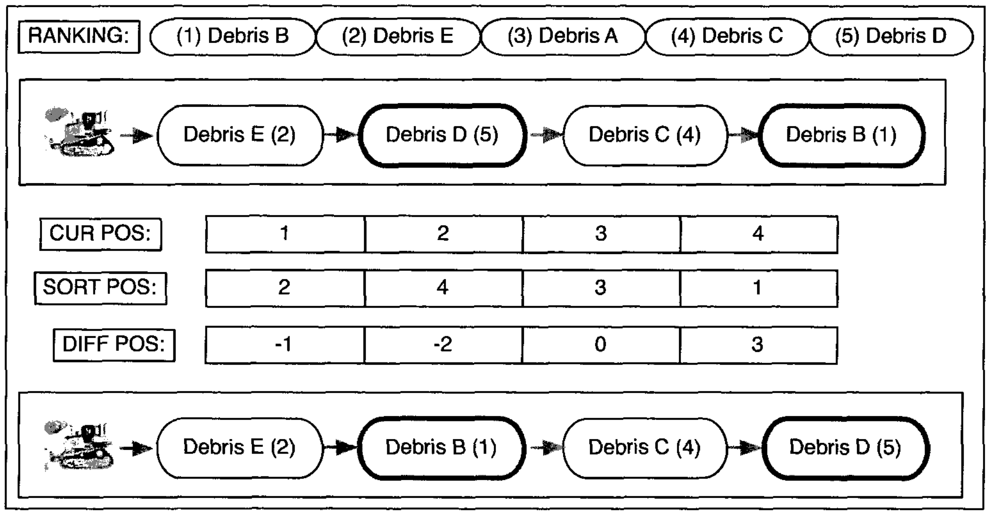


Figure 5.3: Example of the SmartSwap mutation operator. The ranking for the five constraints is shown at the top, with the current constraint sequence for the selected bulldozer below. Intermediate computations are below that, and at the bottom is the result of the SmartSwap operator, where the constraint in the red oval has been swapped with the constraint marked in blue.

This ranking data is used in the SmartSwap function, which acts on a single agent's sequence without considering agents in tandem. The goal of our SmartSwap algorithm is to take the two positions in the agent's sequence that are most out-of-place in terms of the rankings and swap them. To illustrate, consider Figure 5.3. In a SmartSwap a single agent is randomly selected for mutation. The first step the operator takes is to associate each member of its current sequence with its current rank – indexed by position – in the ranking list. Constraints that do not appear in the ranking list are accorded a uniform high value. A copy of the current sequence is then sorted by order of its ranking. The position in this sorted list becomes the sorted position, shown in Figure 5.3 as the "SORT POS". A third number is calculated for each entry in the sequence - the difference between the current position "CUR POS" and the sorted position. These values are shown in "DIFF POS" – negative values represent entries that are earlier than they should be as indicated by the ranking, zero values are properly positioned, and positive values should be earlier in the sequence. The SmartSwap operator then determines the most negative and the most positive position, and swaps those entries in the sequence. In Figure 5.3 this results in Debris Pile B, the most positive value, being swapped with Debris Pile D, the most negative value, producing the sequence shown in the bottom of the figure.

This SmartSwap algorithm has some caveats and limitations. First, like all heuristics it is not guaranteed that all SmartSwaps will improve the schedule quality, especially when the requirements of several fire trucks are considered in tandem. For example, consider a case where a bulldozer is assigned to two debris piles that lie on two different fire trucks routes, one of which is very close to the bulldozer and one which is far away. Even if the distant fire truck reaches its debris first, it may be better in terms of system efficiency to have the bulldozer clear the closer debris first rather than travel to the distant debris pile, clear it, and then travel back. The SmartSwap doesn't fully sort the sequence on the basis of the ranking, but instead makes the single swap which addresses the worst discrepancy in terms of the ranking. If SmartSwaps were repeatedly applied the end effect would be to fully sort the list according to the ranking. Second, the SmartSwap operator only acts on a single agent's sequence, meaning that it will not rectify inefficiencies such as all the earliest ranked debris piles being assigned to a single bulldozer. While it may not improve solutions in all cases, it can potentially result in speeding the discovery of high quality solutions.

5.1.7 Algorithm practicalities

Given this basic algorithm, there are a number of parameters to determine in order to ensure that the genetic algorithm will produce high quality solutions. The two most crucial parameters are the population pool size and the number of generations that the optimization will run. First we consider the number of iterations. Our approach in practice functions somewhat differently than traditional genetic algorithms, due mostly to how the space of fire truck routes is searched. When genomes are initialized fire truck routes will generally not be extensive enough to result in high quality solutions, as they represent only a shortest path to a single fire. Only mutations and crossover in subsequent iterations will serve to make the routes extensive enough to really search the full route space. Thus it is essential that many iterations be run to allow for search of the route space. In terms of population, the population must be large enough to maintain diversity over many iterations. Converging too quickly can result in being caught in local maxima. Increasing the size of the population and the number of iterations will increase the computation time requirements of the algorithm.

Our algorithm has many different parameters, the settings for each of which can have an effect on the quality of the solutions determined. Making parameter tuning especially difficult is the fact the genetic algorithm is a randomized approach: different runs of the algorithm with the same parameters can produce somewhat different results. Thus when changing a parameter it becomes difficult to determine whether or not the new setting improves on the old. Another complicating factor is that, as we show in the results section, optimizations take a good deal of time, especially when run on a set of domain instances. As runs of the algorithm take a good deal of time, any sort of automated search of the parameter space is likely to be impractical. Finally, altering domain parameters used to produce the domain instances may require additional genetic algorithm tuning. The difficulties of parameter tuning are one serious drawback of the GA approach.

Our approach to parameter tuning has been largely ad-hoc, following a few basic intuitions. Setting parameters so that alterations are relatively small and infrequent seems to improve performance. This means setting low values for the non-heuristic mutation percentages and limiting crossover to single fire truck and bulldozer pairs when it occurs. Adding in substantial numbers of SmartSwaps can improve performance, especially in domains with many constraints. Overall, despite the large space of possible parameters, performance seems robust to small changes in parameters. The parameter settings we use in experiments are shown in Table 5.1, unless otherwise indicated:

Our current implementation of the algorithm is not parallelized, running within a single thread, but the algorithm would be easy to alter to obtain substantial speed ups through running on multiple processors or computers. By far the most computationally intensive part of the algorithm is the simulation evaluation; conveniently, this would be an easy place to parallelize the computation. Parallelization for evaluation would involve writing a dispatching algorithm that would either spawn threads for evaluation on multi-processor machines or use network code to send genomes and world descriptions to other computers to run one or more simulations; evaluation results could then be transmitted back. The simulations are completely independent, and thus provide a natural point for parallelization. Once the evaluation for a given generation was complete the results could be centralized again for the relatively less expensive operations of selection, crossover, and mutation. Implementing this parallelized version of the GA approach is left for future work.

5.2 Experiments and evaluation

In this section we offer our most comprehensive evaluation of the relative performance of our full suite of approaches in a simulated disaster response domain. We test with three different genetic

SELECT_TOURNAMENT	.6
SELECT_HALL_OF_FAME	.3
SELECT_NEW__GENOME	.1
CROSSOVER_PROB	.3
NUM_FIRETRUCK_TO_CROSSOVER	1
FT_NODE_DOANYMUTATION	.05
FT_MUTATION_APPEND	.25
FT_MUTATION_REDUCE	.25
FT_MUTATION_LENGTHEN	.25
FT_MUTATION_REDIRECT	.25
FT_WORK_DOANYMUTATION	.05
FT_WORK_SWAP_PER	.05
FT_WORK_ON_PER	.9
FT_WORK_OFF_PER	.1
BD_DOANYMUTATION	.15
BD_MUTATION_INTRA_SWAP	.6
BD_MUTATION_INTER_SWAP	.2
BD_MUTATION_ASSIGNED	.2
BD_NUM_SMARTSWAPS	10

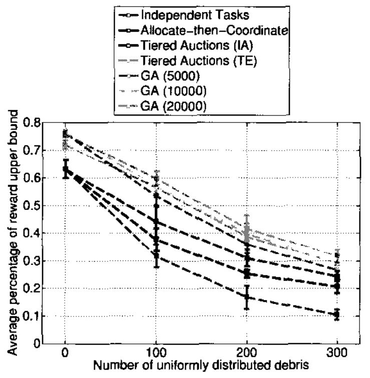
Table 5.1: Default parameters for the genetic algorithm

algorithm population/iterations pairings - 5000 population, 200 optimization generations (GA-5000); 10,000 population, 400 generations (GA-10000); and 20,000 population, 400 generations (GA-20000). Using three different parameterizations will serve to give an idea how long the genetic algorithm takes to determine solutions of various levels of quality.

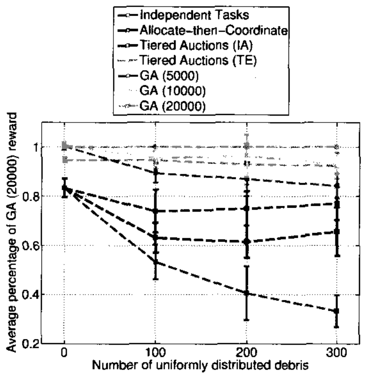
In this section in addition to testing with fixed team sizes and numbers of tasks while varying intra-path density we will also test with varying numbers of bulldozers and varying time requirements for addressing fires.

5.2.1 Varying intra-path density

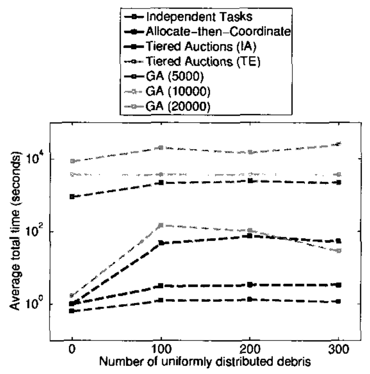
For this set of experiments we use the static domain instance set up as described in Chapter 3 to explore the effects of varied intra-path density on approach performance.



(a) Average reward achieved by approaches as a proportion of an upper bound of available reward, where the bound is the summed reward of all issued tasks if they were addressed immediately. Standard deviations are shown as error bars.



(b) Average reward achieved as a proportion of the reward achieved by the GA-20000 approach on a per trial basis.

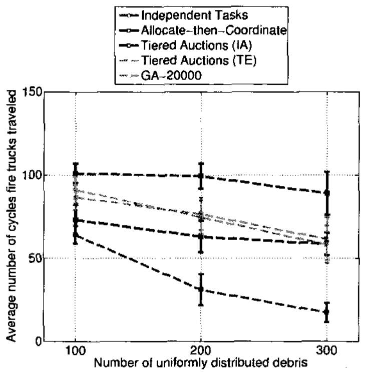


(c) Average total time taken to produce coordination solutions. Trials were all run in a single thread and allowed 100% of a CPU on a quad-core Intel Xeon 3.8 GHz CPU.

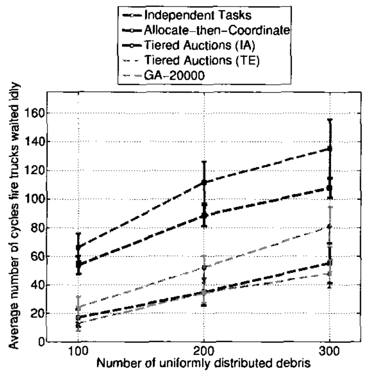
Figure 5.4: Approach performance in a simulated disaster response domain with varied densities of intra-path constraints

We illustrate solution quality in Figures 5.4(a) and 5.4(b). Figure 5.4(a) shows reward in proportion to an absolute upper bound on available reward. Performance differences are clearer in Figure 5.4(b), where we plot how the approaches perform as a proportion of the reward achieved by the best performer, GA-20000. At the highest population level, genetic algorithms outperform the TA-TE approach; the GA-20000 approach achieves an average of 7% more in terms of reward in domains with intra-path constraints. The GA-10000 parametrization achieves roughly similar reward to that achieved by the TA-TE approach, and the smallest parametrization of genetic algorithms, GA-5000, achieves an average of 7% less than TA-TE. All the GA parametrizations outperform the approaches with instantaneous task allocation.

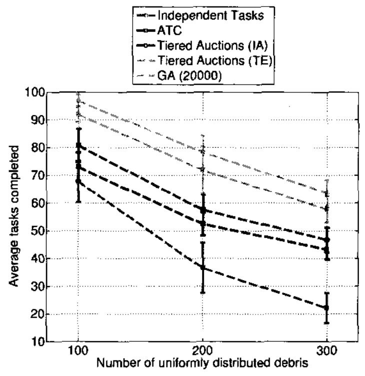
Total time taken to produce coordination solutions is shown in Figure 5.4(c). At all parametrization levels the GA approaches use at least an order of magnitude more computation time than the TA-TE approach, with the GA-20000 approach requiring two orders of magnitude more computation. When computation time is considered, the more heuristic, market-based approaches are shown to offer a good balance of performance and computational efficiency.



(a) Average number of cycles each fire truck spent traveling between task locations



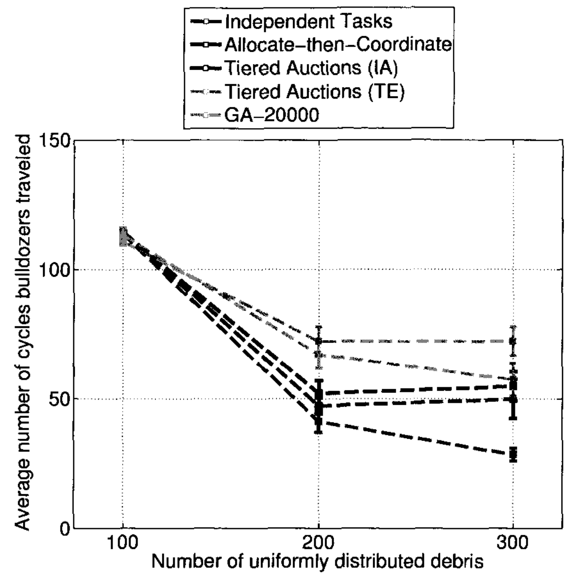
(b) Average number of cycles each fire truck spent waiting at debris piles sites for those debris piles to be cleared during each trial.



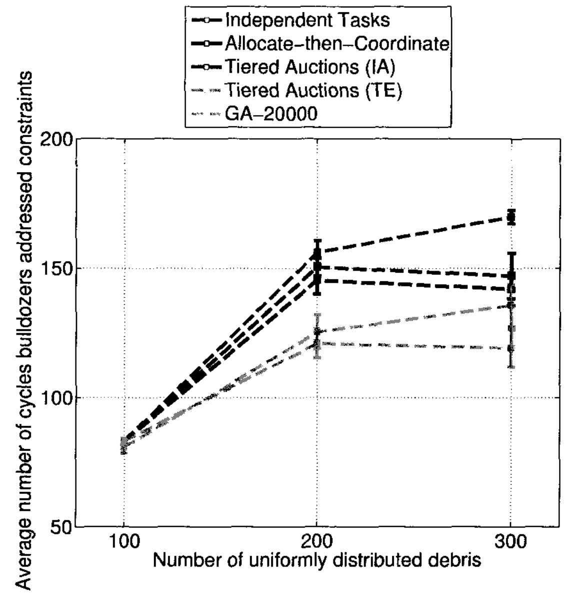
(c) Average number of fire tasks out of the 100 issued that were addressed before reward values reach zero

Figure 5.5: Fire truck behavior produced by approaches in a simulated disaster response domain with varied densities of intra-path constraints

In Figures 5.5 and 5.6 we investigate some of the differences in agent behavior that results from using different approaches to coordination. This examination is not meant to comprehensively account for the performance gaps between approaches, but to quantify some of the sources for the gaps in terms of agent behavior. Figure 5.5(a) shows the amount of time on average each fire truck spent traveling between task locations in domain instances with varying intra-path constraints, and Figure 5.5(b) shows the average time that truck agents spend waiting idly for constraints to be satisfied. In general delay at constraints sites should be avoided whenever possible, and unnecessary travel time should be limited. In the independent tasks approach fire trucks spend the least time traveling and the most time waiting idly; the short travel time is due to the fact that agents address relatively few fires, as shown in Figure 5.5(c), and take shortest paths to reach their assigned fires even if faster paths may be available. The long delay at constraint sites is the product of using a computationally inexpensive implicit coordination mechanism that does not reason about intra-path constraints. At the higher constraint levels agents spend so much time waiting for constraints to be satisfied that they can only reach a few tasks before task rewards reach zero. Agents coordinated using the ATC approach spend substantially more time traveling and less



(a) Average number of cycles each bulldozer spent traveling between constraint sites



(b) Average number of cycles each bulldozer spent satisfying constraints

Figure 5.6: Bulldozer behavior produced by approaches in a simulated disaster response domain with varied densities of intra-path constraints

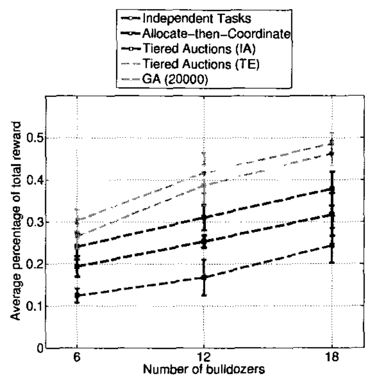
time waiting for constraints to be addressed, though the delay at constraint sites is still more substantial than any of the approaches that consider intra-path constraints during task allocation.

The distinctions between the TA-IA, TA-TE, and GA approaches are considerably more nuanced. In general in order to maximize solution quality we would like approaches to produce behavior where many tasks are addressed with a mininum of travel time and delay. Using the TA-IA approach results in by far the greatest travel time. This approach uses single task allocations without reasoning about clustering tasks or routing in order to opportunistically address tasks. This leads to inefficiency as agents do not reason about sequencing to limit travel time and increase productivity. A strength of the TA-IA approach is that delay is almost as low as in the GA approach - the combination of single task allocation and reasoning about constraints during allocation means that constraint addressing resources are not overexploited even at high constraint density levels. The TA-TE approach is relatively efficient in terms of travel time, as clustering and opportunistic path planning equip agents to reason about taking routes that efficienctly reach a number of task locations. The level of delay incurred is relatively high when compared to the TA-IA and GA approaches, however, with the delay gap increasing with the number of constraints. This is primarily due to overexploitation of bulldozer resources and increases in bulldozer travel time, factors which we discuss in more detail later in the section. The genetic algorithm approach is efficient both in terms of travel time and in terms of delay, traveling roughly the same amount as the TA-TE approach while incurring similar delay to the TA-IA approach. Figure 5.5(c) shows that the GA approach completes more tasks than any other approach, which is especially notable given that both delay and travel time are lower than in the other approaches. While the behavior produced by the GA approach is more efficient by a number of metrics, this added efficiency still only yields a modest 8% performance gain in terms of solution quality over the TA-TE approach, illustrating that more comprehensive search results in diminishing returns over our market-based approaches.

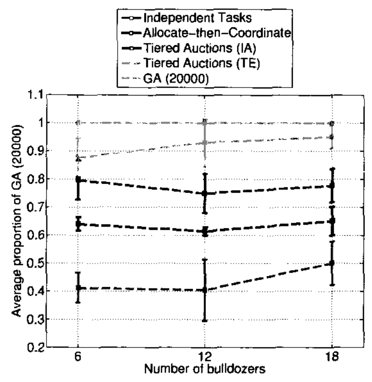
Fire truck behavior is only one important aspect of understanding the performance differences between approaches. In Figure 5.6 we explore bulldozer behavior. Figure 5.6(a) shows the time each bulldozer spent traveling between constraint sites on average, and Figure 5.6(b) shows the time each bulldozer spent addressing constraints. In the independent tasks approach constraint satisfaction duties are assigned entirely on the basis of minimizing travel time, resulting in relatively efficient behavior in terms of maximizing the speed with which constraints are reached and addressed. While this behavior is efficient in terms of speedily addressing all constraints, addressing the particular constraints required by fire trucks at particular times yields better performance in terms of solution quality. The TA-IA approach is somewhat more efficient than the ATC approach in terms of bulldozer behavior, as considering routing during allocation leads to fewer instances where tasks are allocated that require many distant constraints to be addressed. The TA-TE approach is the least efficient in terms of bulldozer behavior - bulldozers tend to travel long distances and spend less time addressing constraints. This is primarily due to the greedy allocation scheme for bulldozers combined with the longer, time-extended routes selected by fire trucks in the approach that require that distant constraints be addressed. The genetic algorithm improves substantially in terms of bulldozer efficiency over the TA-TE approach, resulting in substantially less travel time especially at the 300 constraint level. The GA can more comprehensively assess the benefits of different allocations, sequences, and routes, limiting unnecessary bulldozer travel. Improving the efficiency of constraint-addressing allocation within the time-extended market-based approach is an important matter for future work.

5.2.2 Varying number of bulldozers

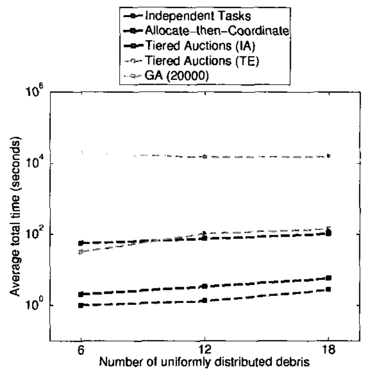
Varying intra-path constraint density is one method we can use to get an idea how a more constrained domain affects performance, but another interesting variable to examine is the number of constraint-addressing agents. As the number of constraint-addressing agents drops in a domain with a fixed number of constraints, constraint-addressing agents become more and more of a scarce resource. In Figures 5.7(a) and 5.7(b) we show the performance of the different approaches in domains with a fixed 200 debris constraint density with three different numbers of bulldozers: 6, 12, and 18. We test with only the best performing GA approach, the GA-20000 parametrization. All other domain parameters were the same as described in Chapter 3.6.



(a) Average reward achieved by approaches as a proportion of an upper bound of available reward, where the bound is the summed reward of all issued tasks if they were addressed immediately. Standard deviations are shown as error bars.



(b) Average reward achieved as a proportion of the reward achieved by the GA-20000 approach on a per trial basis.



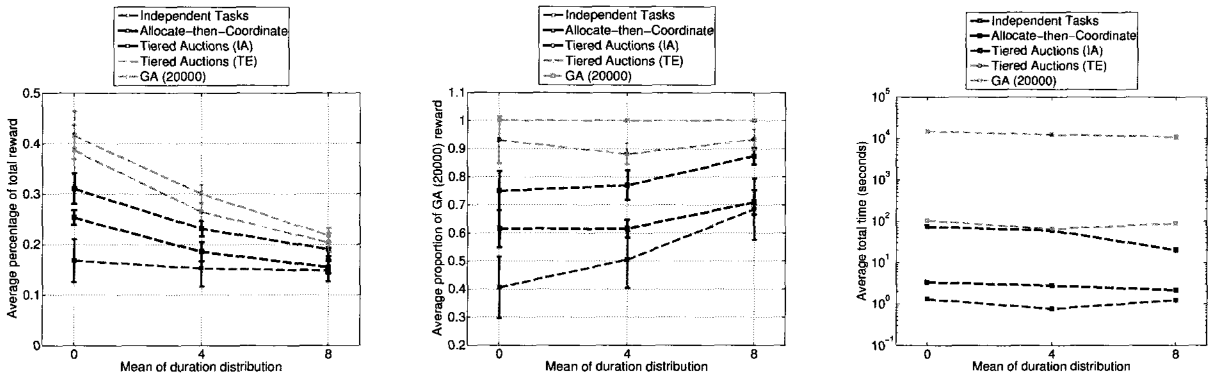
(c) Average total time taken to produce coordination solutions. Trials were all run in a single thread and allowed 100% of a CPU on a quad-core Intel Xeon 3.8 GHz CPU.

Figure 5.7: Approach performance in a simulated disaster response domain with 200 debris pile precedence constraints with varied numbers of bulldozers

The reward received by using the three instantaneous allocation approaches generally improves linearly with the number of bulldozers, and the gaps remain fairly constant in absolute terms. However as a proportion the relative performance of the minimum time approaches compared with the TA-IA approach improves as bulldozer number increases. This is roughly what we would expect, as in cases where there are fewer constraint-addressing resources available considering their actions during allocation and route planning becomes increasingly important. Between the two tiered auction approaches the performance gap is narrowest at six bulldozer setting with only a 10% improvement, increasing to more than 20% at the 12 and 18 levels. At the lowest bulldozer number level, the issues that the TA-TE approach may have in allowing agents to over-exploit bulldozer resources may become more of a factor; tuning the NUM_DEBRIS_CONS parameter could potentially improve performance. The gap between TA-TE and the GA-20000 approach is widest in proportional terms at the 6 bulldozer level, with a 14% gap that narrows to 8% and then to 5% at the 18 bulldozer level.

Figure 5.7(c) shows average total computation time for each of the settings. The time requirement for the market-based approaches all increase with the number of bulldozers - largely because more constraint-addressing agents must determine bids in constraint sub-auctions. TA-TE uses slightly less computation than TA-IA at the 6 bulldozer level, but somewhat more at the other levels. The GA-20000 approach uses roughly the same computation at each of the three levels, and continues to require two orders of magnitude more computation than any of the market-based approaches.

5.2.3 Varying task time requirements



(a) Average reward achieved by approaches as a proportion of an upper bound of available reward, where the bound is the summed reward of all issued tasks if they were addressed immediately. Standard deviations are shown as error bars.

(b) Average reward achieved as a proportion of the reward achieved by the GA-20000 approach on a per trial basis.

(c) Average total time taken to produce coordination solutions. Trials were all run in a single thread and allowed 100% of a CPU on a quad-core Intel Xeon 3.8 GHz CPU.

Figure 5.8: Approach performance in a simulated disaster response domain with 200 debris pile precedence constraints with varied fire durations

The final variation we will test in static domain instances is to vary the time required to address domain tasks - for the disaster response domain this value will consist of the amount of time after a fire truck has reached a fire location that it takes to extinguish the fire. In this set of experiments we tested with three different fire durations settings - one in which fires had zero duration, one where the fire durations were drawn from a normal distribution of $N(4,2)$, and another where the fire duration was drawn from a normal distribution with $N(8,2)$. We test with 12 bulldozers and 200 debris pile constraints.

Performance results are shown in Figures 5.8(a) and 5.8(b). Increasing the duration of tasks has a severe affect on the performance of all approaches except the independent tasks approach. The performance gaps between all the approaches narrows considerably as the duration increases. Making good choices about what constraints to address and what routes to take becomes an increasingly minor consideration, and even the best approaches cannot possibly achieve high reward. If we assume that all fires have a duration of 8, and that all fires reach zero reward at time 120, then the three fire trucks can only address a maximum of 15 fires each during the trial even if travel time or delays associated with constraints; thus its unlikely that even an optimal solution will achieve 50% of the available reward. Though the performance gaps shrink, our approaches that more fully consider the coordination space continue to outperform the approaches that consider less of the problem space.

Computation time is shown in Figure 5.8(c). TA-TE required more computation than TA-IA at all duration levels, and substantially more at the 8 duration level. As fewer tasks could be addressed, the number of auctions held in the instantaneous allocation mechanisms decrease substantially at higher duration levels. GA computation requirements remain unchanged, as the primary component to the GA is simulation time.

5.3 Summary and discussion

In this chapter we described our genetic algorithm method for full coordination space search in static domain instances. The GA approach represents a fundamentally different style of approach than the market-based approaches used in the previous two chapters, eschewing a more heuristic, greedy approach for a less heuristic search based on an evolutionary method. The genetic algorithm searches in the full space of potential coordination solutions, running simulations to determine coordination solution performance given the interactions between agents. Each genome in the GA represents a full coordination solution to a given static domain instance, including representations of the routes that task-addressing agents will take through the environment. Task-addressing agent routes are an important component of the coordination solution, and must be represented. In fact, rather than explicitly representing the full hierarchical coordination space in the form described in Chapter 1.2.2, where task allocations and sequences are explicitly considered, we use a route-based formulation for task-addressing agent schedules. In this formulation task allocation and sequencing are only determined during solution simulation.

Our genetic algorithm initially generates largely random coordination solutions, using simulation to evaluate the solutions. Over many generations, preferential selection on the basis of solution quality channels the search towards higher quality solutions; largely non-heuristic operators for crossover and mutation, which make random alterations without considering or not the alterations will improve performance, serve to search the space of possible coordination solutions. We do employ two primary heuristic methods within the genetic algorithm, both associated with adapting constraint assignments and sequences. The first heuristic method is used during initialization to minimize travel costs for constraint-addressing agents; the second heuristic method is a mutation operator that uses the same insight we used in formulating the earliest completion time heuristic to preferentially perform sequence swaps that minimize delay.

This chapter also presented our most comprehensive evaluation of the suite of approaches in different variations of static domain instances of the disaster response domain. We summarize relative strengths and weaknesses of the approaches in Table 5.2. In addition to varying intra-path density, we also varied the number of constraint-addressing agents and the duration of tasks. The genetic algorithm parametrization that employs the largest population size and number of

Approach	High constraint density	Limited constraint-addressing resources	High task durations	Little time for planning
Ind. Tasks	−−	−	+	++
ATC	−		−	+
TA-IA		+	+	
TA-TE		−	−	−
GA			−	−−

Table 5.2: General guidelines on the strengths and weaknesses of the approaches in static domain instances with different characteristics. Strengths are denoted with the + symbol, and weaknesses with the − symbol, with particular strengths or weaknesses doubling these labels.

generations for search consistently outperforms the TA-TE approach across all tests, showing that full coordination search can outperform our highly heuristic approaches if given sufficient time. Parametrizations of the GA with smaller population sizes and numbers of generations either achieve equivalent or less reward than the TA-TE approach on average. All parametrizations of the genetic algorithm were shown to require at least one or in the case of the best performing parametrization two orders of magnitude more computation time than the TA-TE approach. These results indicate that while full coordination search can potentially outperform the heuristic approaches, the gains are potentially minimal and the computational requirements for doing so are large. This supports the idea that the tiered auction approaches offer a good balance of performance and minimal computation requirements.

The hierarchy of performance remained static over the different experimental conditions: independent tasks < ATC < TA-IA < TA-TE < GA-20000. The gaps between the different approaches varied substantially however. Performance gaps between approaches are a product of a number of different factors. Each domain instance has some portion of the available reward which is relatively easy for agents to obtain - tasks that are near agents and require passage through a minimum of constraints. Even approaches that consider only a small part of the coordination space, such as the independent tasks approach, can obtain the bulk of this award. At the other end of the spectrum are tasks that are extremely difficult to address - lying far away from agents and requiring that many constraints be addressed to allow passage. Even in the best of circumstances such tasks may not yield much reward, as the task value may decay substantially by the time any agent can reach the task. In some scenarios the differences between an approach that can obtain only easy reward and even an optimal approach, that obtains the maximum possible reward given the domain environment, may be small. Whether or not using a more computationally expensive approach justifies the payoff in terms of improved solution quality will be a product of a number of different scenario factors.

A performance gap that would be of considerable interest to explore is the gap between our best performing approach - the genetic algorithm approach with sufficiently high population and generations - and an optimal coordination solution. As described in Chapter 2.3.3, there are approaches that could potentially determine optimal solutions to static domain instances with intra-path constraints, and comparison could be made for small domain instances. Unfortunately our domain instances are sufficiently large in terms of the number of agents, tasks, constraints, and potential routes such that determining provably optimal solutions will potentially be intractable. Quantifying this performance gap remains an area of future work. Another area of future work is determining if employing more heuristics or making some other alteration to our full coordination search algorithm could achieve even greater performance or could achieve the same performance levels more quickly. This is another area of future work we will consider at more length in Chapter 8.

Chapter 6 Coping with uncertainty

In the previous chapters we introduced our suite of approaches within the context of static domain instances, where it is assumed that the domain state is perfectly known and unchanging. The world is not static, however, and every system that involves agents must confront uncertainty, as there are few if any situations where knowledge is perfect and domain requirements unchanging. Instead, new domains requirements will be discovered or generated, new environmental conditions sensed, or the composition of the team may change due to agent incapacitation. Uncertainty can have severely deleterious effects on the quality of a coordination solution. A set of agent plans conceived under one set of assumptions may vastly decrease in quality if those assumptions incorrectly describe the world state. A further complication from a planning standpoint is that in many domains uncertainty is not the exception, but is instead the rule – in such domains either the perception of the world state or the actual domain requirements can be assumed to be in a constant state of flux.

Each time the perceived world state changes, whether due to new environmental data, new domain requirements, or reduced agent capabilities, the shape of the coordination space changes, potentially altering the quality of the currently adopted coordination solution. In dynamic domain instances the coordination process can no longer be one of generating a solution and expecting agents to follow those solutions perfectly; rather, determining the coordination solution necessarily becomes a dynamic process, where the solution must be frequently adjusted or reformulated to accommodate the newly discovered information. Interspersing planning and execution becomes essential, as only during plan execution will new information come to light. The more uncertainty that exists in the environment, the greater the emphasis of a coordination approach should be on re-planning in the face of new information rather than on establishing agent schedules ahead of time. The presence of uncertainty in the environment can alter the determination of what time horizons are appropriate for planning, and reacting quickly to changing requirements may become as important to solution quality as searching more fully in the coordination space.

With the exception of the genetic algorithm approach, all the approaches we have presented use planning after execution has begun even in static domain instances, a process we call re-planning. Approaches with limited time horizons will re-plan as a matter of course when some agents have exhausted their schedules, though the longer the time horizon for plans the less frequently such re-planning will occur. Though most of our approaches re-plan regularly as part of the algorithms, in dynamic domain instances approaches may need to take a more intentional stance to re-planning, potentially re-planning not just when agents have exhausted their schedules but in reaction to newly discovered information about the environment. In terms of reactive re-planning – re-planning in response to new information – there are two extremes in terms of extent: full reactive re-planning or no reactive re-planning. In a full reactive re-planning approach, when uncertainty results in any sort of alteration to the coordination space a complete re-planning phase occurs. A complete re-planning represents a search of the new coordination space of the same scope as occurred when forming the original coordination solution. At the other extreme, no reactive re-planning, no considerations are given to reacting to new information - whatever schedules are in place will be followed, with re-planning occuring only when agents' schedules are exhausted. A full reactive re-planning approach will likely require a good deal of computation time, while doing no re-planning obviously will be computationally inexpensive.

A third alternative remains, however; between these two extremes are partial re-planning approaches, where some level of consideration is given to adjusting the coordination solution in response to new information, but less effort is expended than would be associated with doing a full coordination space search. A partial re-planning approach may also take into account the nature of the environmental change, determining whether a higher or lower degree of re-planning is required. What approach makes the most sense for a given domain is a product of the specifics of the approach, the type of uncertainty, and the time available for computation. Approaches that naturally re-plan frequently may not require reactive re-planning, while approaches with longer time horizons will almost certainly need some method to react to new information.

In some cases, the effects of uncertainly may be relatively minor in terms of altering the coordination space or changing the relative quality of the current solution. Minor changes will not affect the ability of the agents to execute their schedules largely as planned, and will not make a current high quality solution a poor one in relative terms. One example of this kind of uncertainty are small changes in the durations required for executing particular actions. If addressing a constraint takes a fraction longer than expected, or it takes an agent slightly longer to traverse a segment than scheduled, the effects on solution quality will be minor - at most a few cycles of delay. Agents can still largely execute their explicitly scheduled plans. For minor durational uncertainty, full re-planning approaches are unlikely to substantially improve performance, and for shorter time horizon approaches not doing any explicit re-planning is probably the best approach. For longer time horizon approaches small durational changes can accumulate over time to cause more substantial delays later in schedules; in such domains periodic full re-planning or partial re-planning approaches may be needed.

Uncertainty associated with dynamic task issue can potentially also fall under the category of minor uncertainty. New task issue does not affect agents' ability to execute their schedules as planned - the fact that new fires are discovered, for instance, does not introduce additional constraints or reduce agents' ability to complete their explicitly scheduled plans. At the same time, solutions conceived in a coordination space involving one set of tasks may perform relatively poorly in a new coordination space including newly issued tasks. The extent to which reactive re-planning in response to dynamically issued tasks can improve performance is a function of the planning horizon, the frequency of task issue, and the characteristics of the task being issued. In domains where task issue is infrequent and new tasks are of similar importance and urgency to the previously existing tasks, reactive re-planning may be of limited benefit. However, consider a situation where a task is issued that is orders of magnitude more important and urgent than any task being addressed in the current coordination solution - for instance, a huge fire at a hospital when all previous fires were at houses. In this scenario whatever the planning horizon not using reactive re-planning may result in an unacceptable loss of solution quality, as any coordination solution that does not involve some agent immediately moving to address the new task will be relatively poor. If such abrupt changes are an expected part of a domain, reactive re-planning can potentially benefit all approaches.

While domains with dynamic task issue may or may not require re-planning to achieve high reward, in some domains the need for some form of reactive re-planning is unequivocal, especially for explicitly scheduled approaches. One primary class of domains in which re-planning is essential for quality performance are domains where explicitly schedules solutions may be impossible to execute as scheduled given new information. For instance, in the disaster response domain it may be the case that only a partial list of debris pile locations is known, or that buildings can crumble and form new debris piles during execution. If a newly discovered constraint location blocks the intended route of a fire truck, then its schedule will no longer be accurate. If no extra constraint-addressing assistance is provided, the solution becomes implicitly, rather than explicitly, scheduled. Similarly, if addressing a task or satisfying an intra-path constraint was thought to require some number of agents or capabilities and turns out to require some other configuration, then the intended schedules will no longer be accurate. If an explicitly scheduled coordination approach has no mechanism for repairing solutions that make schedules inaccurate and such events regularly occur the effects on solution quality can be drastic. Domains where there are frequent and large discrepancies between scheduled durations and actual durations can also have a similar requirement that the solution must be re-evaluated to maintain good performance. Maintaining explicitly scheduled coordination in domains with this kind of uncertainty can become difficult.

For such domains there are two general approaches that can be taken to maintain explicitly scheduled coordination - schedule repair and coordination re-consideration. In schedule repair agents largely maintain their schedules and assignments, recruiting new agents as needed to produce accurate explicit schedules that account for the new information. This approach can be considered a partial re-planning approach. In coordination re-consideration agents largely abandon their current schedules, substantially searching the coordination space for new solutions - this is akin to full re-planning. A final possibility for domains with high uncertainty to move away from an explicitly coordinated approach to one that is more implicitly coordinated. We leave this third possibility for future work.

In this chapter we divide our inquiry on uncertainty into two sections. In the first section we focus on dynamic task issue as a case study in relatively minor uncertainty. We discuss potential adaptations of our approach to support reactive re-planning and evaluate the performance of our adapted approaches on a disaster response domain with dynamic task issue. In the second section we consider a domain where constraint locations are largely unknown as a case study in coping with uncertainty that significantly impairs the ability of explicitly scheduled coordinated plans to be executed as intended. We discuss potential approaches for both schedule repair and schedule re-evaluation, focusing particularly on adapting the tiered auction with instantaneous assignment approach.

6.1 Strategies for dynamic task issue

In this section we focus on strategies for coping with dynamic task issue in disaster response. We primarily focus on domain instances with relatively frequent task issue, but where new tasks are approximately equivalent to existing tasks in terms of importance and urgency. The fact that new task issue is frequent means that reactive re-planning can be of substantial benefit, as a series of newly issued tasks can substantially alter the coordination space. At the same time, the relative uniformity of task importance and urgency mean that it is not absolutely essential to reactively re-plan to accommodate newly discovered tasks that are orders of magnitude more important or urgent.

As stated in the introduction to this chapter, dynamic task issue does not affect agents' abilities to execute their schedules, whether explicitly or implicitly coordinated. This means that all of our approaches are directly applicable as described for static domain instances - reactive re-planning is not required to maintain explicitly scheduled coordination. All of the approaches except the GA already do some amount of re-planning during execution. When agents have exhausted their schedules, the newly arrived tasks can be considered for allocation using the methods we have already presented. The genetic algorithm approach is unlikely to perform well unchanged in a domain with dynamic task issue, as the full time horizon approach as presented does not have a built-in provision for re-planning. If optimization only occurs at time zero, then the solution will only reflect the set of tasks known to exist at time zero. While the genetic algorithm approach should find a high quality solution given the set of tasks issued at time zero, for dynamic domain instances with frequent issue altering the initial coordination solution to reflect new task issue will likely be requisite for achieving high performance.

While all approaches can be used without alteration in domains with dynamic task issue, reactive re-planning can potentially improve performance, especially for approaches with longer planning horizons and consequently more infrequent re-planning. On the other hand, the more extensive the efforts to reactively re-plan, the greater the computational cost. In fact, in dynamic domain instances the performance improvement associated with reasoning over longer planning horizons may be reduced compared to the gain achieved in static domain instances. For the genetic algorithm approach, for instance, spending substantial time searching in the full coordination space over the full horizon of available tasks may represent a good deal of wasted effort. The coordination space is likely to change substantially as agents execute their schedules, making it unlikely that agent plans made far in advance will continue to be of relatively high quality in the future. Given that time is precious, the right balance to strike between spending more time conducting search during each re-planning phase or re-planning more frequently is an open question. Ideally, approaches would both extensively search in the coordination space and constantly re-plan, but doing so may be prohibitive in terms of computational cost.

In the rest of this section, we focus primarily on reactive re-planning approaches for our longer time-horizon approaches - tiered auctions with time-extended assignment and genetic algorithms. In the next section we briefly describe why such reactive re-planning may not represent a good use of resources in our shorter time horizon approaches. We then focus on different possibilities for reactive re-planning strategies for the longer time horizon approaches. Finally, we present experimental results of approach performance in disaster response domain instances with dynamic task issue.

6.1.1 Beneficial features of short horizon algorithms

In our short horizon approaches – independent tasks, ATC, and TA-IA, which assign a single fire per truck at a time – re-planning occurs frequently as a part of the algorithm, and we consider it unlikely that doing re-planning specifically in response to dynamic task issue will substantially increase performance. Any potential reactive re-planning strategy would function by considering whether or not to change the current assignments of agents. While one can create scenarios where in particular instances it makes sense to interrupt the addressing of one task to have an agent address another, in practice it does not seem to substantially increase performance.

Our short horizon algorithms are already well-suited for operation in dynamic task domains due to two key design features - the use of average reward as the bidding metric and using NUM_-DEBRIS_CONS to restrict the number of intra-path constraints on a path. In Chapter 3 we discussed the reasons for using average reward per cycle as the individual agent metric instead of total reward - this metric becomes even more important when domains include dynamic task issue. The primary reason we use average reward in domains with static task issue is that agent single task bids do not account for the fact that all tasks have decaying reward. In domains with dynamic task issue getting reward as quickly as possible is even more important, as new tasks will constantly be discovered. Assignments made on the basis of maximizing average reward are likely to involve tasks that can be addressed quickly, reducing the time horizon and leading to faster re-planning cycles and greater responsiveness. Similarly, by setting low values for NUM_DEBRIS_CONS we prevent the over-exploitation of bulldozer resources, allowing constraint-addressing resources to be responsive to fire truck requirements associated with newly issued tasks. In our experiments with potentially altering single-task assignments in response to new task issue, computational costs were increased with little or no improvement in solution quality.

6.1.2 Adapting time-extended tiered auctions for dynamic task issue

As is the case in the shorter horizon approach, the time-extended tiered auction approach benefits from a number of design decisions made to enhance performance in static domain instances. This approach also uses average reward instead of total reward as well as employing the NUM_DEBRIS_-CONS parameter. Additionally, limiting the indirection of paths using g-value bounding also reduces the planning horizon in many instances. At the same time, the horizon used by the tiered auction with time-extended assignment approach is substantially longer than that employed in the single-task allocation approaches. With the longer horizon comes an associated lack of responsiveness - if no reactive re-planning is done to accommodate newly issued tasks then the infrequency of the planning intervals can impair performance.

Our initial effort in developing a re-planning algorithm for the TA-TE approach was to explore partial re-planning approaches. Our goal was to design an approach to minimally alter existing schedules and to do so in a manner that required little extra computation. Using minimal computation precluded re-running route search for fire trucks or allocating new debris pile responsibilities to bulldozers. One approach that required little extra computation was to have agents use the opportunistic path planning approach described in Chapter 4.2 to potentially incorporate new tasks into their existing schedules. This approach adds newly issued fires to fire truck schedules' if those fires can be easily reached without requiring passage through unassigned constraints and if doing so will increase the average reward of the schedule. One complicating factor is that it may be the case that more than one fire truck can potentially increase its average reward by addressing particular new tasks. To address this case we take the approach of holding a sequential single-item auction for any newly issued tasks - the agent that can improve their average reward the most by addressing the task is awarded it.

Algorithm 9: Auction approach for potential incorporation of newly issued tasks into existing agent plans

```
Input: $T$: List of new tasks
       $A$: List of fire truck agents
Result: New tasks may be incorporated into agents' schedules
1 foreach $t$ in $T$ do
2     $maxBid \leftarrow 0$
3     foreach $a$ in $A$ do
4         $bid \leftarrow$ BidOpportunisticallyForNewTask($t$,$a$)
5         if $bid > maxBid$ then
6             $maxBid \leftarrow bid$
7     if $maxBid > 0$ then
8         IncorporateNewTaskIntoSchedule($maxBid.agent$,$t$)
```

The algorithm for the auction using opportunistic path planning is shown in Algorithm 9. The function `BidOpportunisticallyForNewTask` will only consider adding the particular new task to the current schedule - no additional tasks are considered. If the task cannot be opportunistically altered, either due to being separated from the route by debris or requiring a detour greater than the parameter OPP_MAX_DISTANCE then the bid will be 0. If the task can be added but lowers the agent's scheduled average reward, the bid will be negative. If adding the task improves an agent's scheduled reward, the bid will be positive. Each new task will only be assigned if some agent places a positive bid for the task. If more than one agent can improve its reward by adding the task to its schedule, the task is awarded to the agent that gains the most from adding the task. Each task is considered individually by each agent in this process.

In practice this approach can yield substantial improvements for long horizon approaches. One scenario that occurs with reasonable frequency is that fire trucks will be sitting idly for considerable durations waiting for particular debris piles to be cleared. During this time if a task arrives that is near the idle agent then it will almost certainly improve the quality of the coordination solution to have the agent address the new task rather than sitting idly. By setting high values for the distance limiting parameter OPP_MAX_DISTANCE the opportunistic path planning will consider adding fires a larger distance from the trucks, though the fires will only be opportunistically added if doing so represents an improvement in average reward. This approach is also quite fast, as the most computationally expensive component of opportunistic path planning process is a simple distance-limited breadth-first-search. Additionally, in many cases using this partial re-planning approach can actually decrease the total computational cost. Allocating new fires using opportunistic path planning is relatively computationally inexpensive, while having to allocate new fires during the cluster auctions is more expensive. Thus supporting the incorporation of new fires into existing schedules results in fewer tasks to consider using expensive methods.

We did some experimentation with more extensive partial re-planning as well as full re-planning approaches. One approach we took was to periodically discard the current coordination solution and to re-plan from scratch. Another approach we tried was to enable agents to periodically assess whether or not addressing a different cluster would improve the agents' average reward. In neither case did our efforts result in consistent solution quality improvements, and the computational costs of the approaches were substantial. Continuing this line of experimentation is a potential avenue for future work.

6.1.3 Adapting genetic algorithms for dynamic task issue

Of all our approaches genetic algorithms requires the most modification to cope with dynamic task issue. Dynamic task issue presents a serious challenge to genetic algorithms in terms of tractability. Genetic algorithms search in full coordination space, but dynamic task issue means that the coordination space will be in flux. One potential approach to the changing coordination space is to repeatedly run genetic algorithm optimization exactly as described in Chapter 5 whenever new tasks are issued. While this approach should achieve high quality results, we already showed in static domain instances that the GA approach takes orders of magnitude more time to improve on the solutions produced by the heuristic techniques. Repeated full optimization cycles could potentially increase the computational requirements by several more orders of magnitude. Full horizon optimizations also represent a good deal of wasted work, as only a small part of the planned coordination solution will be executed before another optimization cycle occurs. Our goals in adapting genetic algorithms to dynamic task issue were threefold - to use full re-planning, but perform it as infrequently as possible and to leverage the results of previous searches as much as possible, to use partial re-planning to supplement full re-planning, and to focus the search on the short horizon parts of the schedule.

We first discuss how we incorporate full re-planning into our GA approach in dynamic task domains. In implementing a full re-planning approach we wanted to leverage previous searches in the hopes that we could use smaller populations and fewer generations during re-optimization to speed search. While new task issue alters the coordination space, the current schedules associated with the previously determined coordination solution still represent a much better start for optimization compared to randomly generated genomes. Thus we created a new method of genome initialization that translates current agent schedules at the time of re-optimization into the sequences used by genomes. Search in the coordination space then proceeds with the entire genome pool initialized from the current coordination solution. The algorithm for full re-planning is identical to Algorithm 8 except that the function `InitializeGenomePool` initializes the entire genome pool with a genome translated from the current coordination solution. Our full re-planning approach for genetic algorithms is to run optimization at time zero using random initialization and then to re-optimize every FULL_REPLANNING_PERIOD cycles using the current agent schedules for genome initialization.

While ideally we would run a full optimization cycle in response to each new task issue, in practice this may be impractical from a tractability perspective. To maintain tractability, the parameter FULL_REPLANNING_PERIOD should be set so that full re-planning occurs as infrequently as possible while maintaining solution quality. In the interest of generating high quality solutions while doing full optimization as infrequently as possible, we sought to employ a partial re-planning strategy to supplement the full optimization approach. Reactive partial re-planning can have two benefits. First, using partial re-planning can increase the responsiveness of the GA approach without requiring more full optimization cycles. Second, partial re-planning can heuristically improve agents' schedules, offering a better starting point for search when full optimization executes. We chose to use the same partial re-planning strategy for genetic algorithms as we used for tiered auctions with time-extended assignment. This approach is capable of making small alterations to schedules to accommodate new tasks, and helps the genetic algorithm approach be responsive in situations where fire trucks can potentially address newly issued tasks while waiting for constraints to be addressed. Using reactive partial re-planning does not replace the need for full re-planning, but can be beneficial in achieving high quality solutions and allowing full optimization to be required less frequently.

Our final adaptation of the genetic algorithm is to focus the search on what actions agents will execute in the immediate future. The reasons for focusing the search are two-fold. First, making good decisions for the immediate future is especially important, as the coordination space is likely to have substantially changed by the time future actions are taken. Second, we actually want to encourage the genetic algorithm approach to be somewhat greedy. Unlike in domains with static task issue, the coordination space being searched at any given time does not reflect the true coordination space, but only the space associated with the set of tasks that are currently known. In static domain instances there is no reason to prefer near-term reward, but in dynamic issue domains it can improve performance for agents to maximize near-term reward with the expectation that new tasks will arrive rather than deferring near-term reward for greater future reward.

We experimented with several methods for focusing search. We initially tried employing a fixed time horizon. In this method rewards obtained after some future time would not be factored into fitness evaluation. This approach performed poorly, as in many cases it makes sense to consider agent actions beyond any fixed horizon, especially in terms of bulldozer deployment. If a fixed horizon is used, beyond the horizon bulldozer actions have no effect on fitness and will be essentially determined randomly. Instead we settled on a standard technique from the Markov Decision Process literature to discount future rewards in the face of uncertainty. In this approach we set a discount factor co-efficient DISCOUNT_FACTOR (d in subsequent calculations) between 0 and 1. When computing fitness values during the `EvaluateState` function we compute the reward received for tasks using the equation $r' = r * (d^t)$, where the scheduled reward is r and t represents the number of cycles from the current time that the task is scheduled to be addressed. For values of d near 1, the discount associated with the time horizon is negligible; as d decreases the discount for future rewards increases steeply. This form of discounting has a desirable effect on the search. If two solutions are equivalent in terms of full horizon reward, using a discount factor will slant the search towards the solution that achieves the reward more quickly - this makes the search somewhat greedier. If two solutions are equivalent in terms of short-term reward, discounting will still give preference to the solution that achieves higher long-term reward. This means that bulldozer actions far in the future will still factor into the fitness evaluation. We settled on the value of $d = .9$ through experimental validation on a number of different domain instances, though tuning for particular domain parameterizations will be required to maximize performance.

6.1.4 Experiments and evaluation

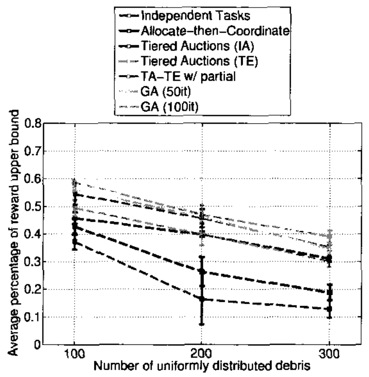
In this section we evaluate the different approaches in a simulated disaster response domain with dynamic task issue. Each dynamic domain instance will have some set of tasks that are issued at time zero and tasks that are discovered during execution. Our approach to dynamic task issue is to suppose that fire issue is the product of a single Poisson process, the standard distribution used in queuing theory to represent stochastic arrival times of independent tasks (Gross & Harris, 1998). The parameter λ for the Poisson process represents the expected rate of task issuance, as governed by the Poisson probability distribution for given $x = 0, 1, \ldots$:

$$f(x, \lambda) = \frac{\lambda^x e^{-\lambda}}{x!} \tag{6.1}$$

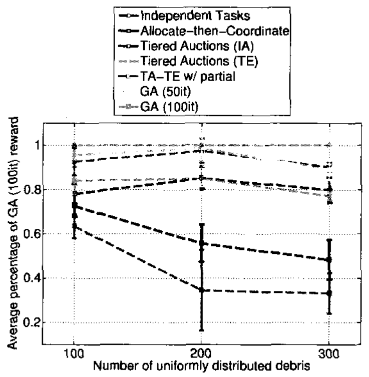
Each cycle the Poisson process is polled using a set value for λ; the output is the number of tasks that were issued that cycle. New tasks will occur with uniform probability across the domain environment. In these trials we set $\lambda = 2$, with 40 initial tasks.

One challenge in evaluating approaches in domains with dynamic task issue is determining the duration of the trial. In domains with static task issue this is not an issue, as after a certain point there is no more reward available, and evaluation can cease. In domains with dynamic task issue, however, new sources of reward are continually made available. Ideally we would run trials until some sort of steady-state performance is reached, as we did in our previous work in disaster response without intra-path constraints (Jones *et al.*, 2007). However, as bulldozers satisfy constraints the domain environment becomes less and less constrained. The approach we take is to arbitrarily limit the term of the trial to 150 cycles, evaluating approaches by how much reward agents operating in the domains can accumulate in these 150 cycles. We keep our approaches agnostic towards time in that knowledge of the trial term does not factor into decision making; at the same time, imposing a limit of any sort will privilege greedy approaches to some extent. Given the settings of 40 initial tasks and a setting of $\lambda = 2$ an average of 340 tasks will be issued in a 150-cycle trial.

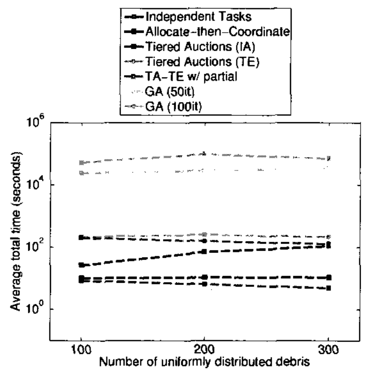
We test using three constraint frequency levels of 100, 200, and 300, running 5 trials at each frequency level. The team consists of three fire trucks and 15 bulldozers. The independent tasks, ATC, and TA-IA approaches are unchanged from those used in static domain instances. The tiered auction approach is unchanged except that we test with two versions - TA-TE with no reactive re-planning, and TA-TE with reactive partial re-planning for new tasks (TA-TE w/ partial). We run with two different parametrizations of the genetic algorithm. The two different parametrizations each use a discount value of $d = .9$, using reactive partial re-planning for incorporating new tasks, and running full re-planning every 8 cycles. The discount factor was determined by experimentation with different settings and selecting the one that consistently lead to the best performance. With a duration between full re-planning of more than 8 cycles the GA approach will not consistently outperform the TA-TE approach even if using partial re-planning. During initial full optimization and each subsequent full optimization the first parametrization, GA-50it, uses a population of 10,000 genomes and optimizes for 50 iterations. The second parametrization, GA-100it, uses a population of 10,000 and optimizes for 100 iterations.



(a) Average reward achieved by approaches as a proportion of an upper bound of available reward, where the bound is the summed reward of all tasks if they were addressed immediately upon issue. Standard deviations are shown as error bars.



(b) Average reward achieved as a proportion of the reward achieved by the GA-100it approach on a per trial basis.



(c) Average total time taken to produce coordination solutions. Trials were all run in a single thread and allowed 100% of a CPU on a quad-core Intel Xeon 3.8 GHz CPU.

Figure 6.1: Approach performance in a simulated disaster response domain with dynamic task issue and varied densities of intra-path constraints

Figures 6.1(a) and 6.1(b) show performance results from our trials with dynamic task issue. We first consider the performance difference between the minimal time approaches and TA-IA. The performance gap is relatively small at the 100 constraint level, but increases considerably at higher constraint levels. At the lowest constraint level none of the approaches will spend much time being delayed at constraint sites, as the 15 bulldozers quickly address constraints no matter which approach is adopted. At the higher constraint levels in the minimal time approaches more delay will occur at constraint sites as constraints are not considered during assignment. In domains with dynamic task issue, such delays become potentially very costly, as new tasks appear frequently. If agents are delayed for substantial periods at constraints they cannot be involved with addressing newly issued tasks.

The performance of TA-TE without re-planning and TA-IA are virtually identical - the greater responsiveness of the TA-IA approach balances with the longer planning horizon used in the TA-TE approach. When partial re-planning is incorporated the TA-TE approach outperforms TA-IA; the performance gap is widest at the 100 constraint level, and narrows at the higher constraint levels. This largely mirrors the results from static domain instances, except that the performance gaps are on average smaller than those found in static domain instances - 16% versus 24% in static domain instances. Even when TA-TE uses partial re-planning, the instantaneous allocation approach has higher reactivity to new information with its shorter natural re-planning cycle. The benefits of planning further into the future are decreased by uncertainty, and especially if the solution is not adapted to better fit changing perceptions of the world state. The performance differences between the TA-TE with partial re-planning approach and the most computationally expensive genetic algorithm approach largely mirror those found in the static domain instances, with an average gap of 7% between TA-TE and GA-100it. We again find that the extra computation helps substantially, as the performance of the GA-50it approach is roughly equivalent to the TA-TE approach with partial re-planning.

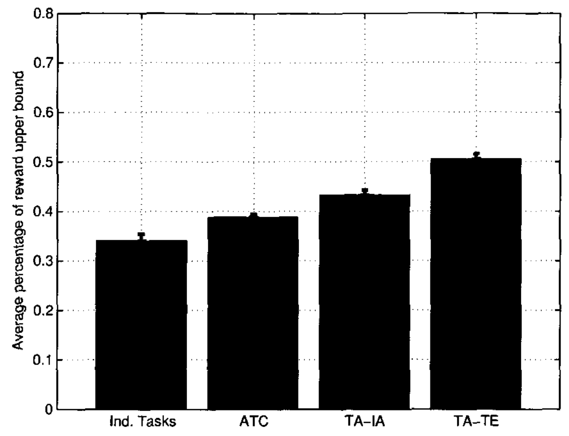
Total required computation time for the dynamic task issue trials is shown in Figure 6.1(c). The differences in computation time between the market-based approaches are roughly similar to those found in static task domains. An interesting note is that using partial re-planning actually considerably shortens the computational requirements of the TA-TE approach. Our method for partial re-planning leads to longer schedules, as agents potentially need to take detours to reach new tasks; longer schedules mean that auctions need to be held less frequently. Furthermore, the partial re-planning approach is a computationally inexpensive way to incorporate tasks into schedules; if partial re-planning is not used there are more unallocated tasks to consider in future cluster auctions. Both versions of the genetic algorithm use between 2 and 2.5 orders of magnitude more computation than the TA-TE with partial re-planning approach. The solution quality achieved by the GA could potentially benefit from using even more search time; we could do full optimization more frequently or use larger populations or optimize for more generations. Even in GA-100it, we use only a population size of 10,000 for 100 iterations, substantially less than the 20,000 population for 400 iterations we use in the static domain instance trials. An approach that frequently fully re-plans and does full coordination search can potentially be both reactive and plan into the future, but the cost will be high in terms of required computation.

Much larger domain instances

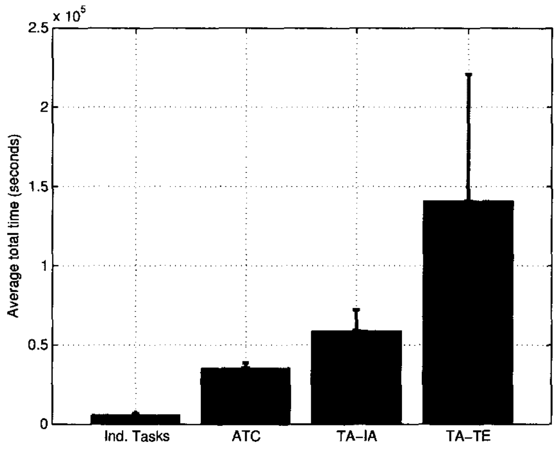
In the trials conducted thus far we have considered varying a number of different parameters from the domain, but all of the trials were conducted in domain instances of roughly similar sizes: 3 fire trucks, 6-18 bulldozers, and 100- 350 fires in a 7-by-5 road network. In this section we consider how the approaches perform in much larger domain instances. First, we expand the domain arena to a 12-by-9 street network, which has roughly four times the area of the 7-by-5 environment. Our team consists of 15 fire trucks and 60 bulldozers. 800 debris pile precedence constraints exist in the domain, with 100 initial fires and an expected rate of fire issue of $\lambda = 10$. We average results over three trials.

We use the same parametrizations for the market-based approaches as described in the last section. Tractability becomes a significant concern in the genetic algorithm approach, as simulating many more agents in a larger environment substantially increases the computational requirements of the approach. Ideally in the larger, more complex environment we would use a larger population size and optimize for more generations to account for the much larger size of the coordination space. However, the genetic algorithm implementation already requires a good deal of computational resources in smaller domain instances. Larger domains mean that simulation is substantially more time consuming, and we run into memory limits even on an expensive server. Making the problem even more difficult for domains with dynamic task issue we should frequently re-optimize to improve performance. In initial testing we determined that a single optimization for a population size of 1000 for 400 iterations required an average of more than 50,000 seconds. To use the same parametrization we used in the GA-20000 approach, with a 20,000 population size for 400 iterations would require around 1 million seconds – over 250 hours – even if memory was no concern. This is without increasing the population size, optimizing for more iterations, or any re-optimization to adapt solutions in domains with dynamic task issue. Due to time constraints we will not attempt to test the GA in this section.

Performance in the much larger domain instance is shown in Figure 6.2(a). The performance hierarchy remains unchanged from smaller domain instances, but the performance gaps are somewhat different. The minimal time approaches perform relatively well, with a gap of 48% between TA-TE and independent tasks. Standard deviations also tend to be relatively small, as the larger domain instances mean that performance across different domain instances is more uniform than in smaller domains, de-emphasizing individual decisions. Computation time is shown in 6.2(b). We first note that all of the computation times for all approaches are much higher than in smaller domain instances - even the independent tasks approach requires over 5000 seconds. There are many agents to coordinate in the domain, and even in the minimal time approaches there is substantial computation required to bid on the high numbers of tasks and constraints. Approaches that were even more efficient in terms of computation than our minimal time approaches could be tailored for these larger domain instances. Comparing the computation times of the three instantaneous alloca-



(a) Average reward achieved by approaches as a proportion of an upper bound of available reward, where the bound is the summed reward of all issued tasks if they were addressed immediately. Standard deviations are shown as error bars.



(b) Average total time taken to produce coordination solutions. Trials were all run in a single thread and allowed 100% of a CPU on a quad-core Intel Xeon 3.8 GHz CPU.

Figure 6.2: The performance of the market-based approaches on substantially larger domain instances.

tion approaches, the gap narrows considerably from that found in previous trials. Using the TA-IA approach only requires about twice the computation time of the ATC approach, a much smaller gap than the factor of almost 20 found in some of the other trials. The TA-TE approach is far more computationally expensive than the other approaches, requiring over twice the computation of the TA-IA approach.

6.2 Coping with other forms of uncertainty

In the previous section we illustrated the performance of both short and long time horizon approaches in dynamic domain instances with uncertainty in the form of dynamic task issue. We also described methods for reactive and full re-planning for longer time horizon approaches. While dynamic task issue can alter the relative value of the currently adopted coordination solution, it is a minor form of uncertainty. Our short horizon approaches perform relatively well without incorporating any reactive re-planning beyond that normally associated with employing short horizons. From the perspective of determining and maintaining high quality explicitly scheduled coordination solutions there are many other forms of uncertainty that are more problematic. Such uncertainty goes beyond minor changes in duration or dynamic task issue to forms of uncertainty that compromise explicitly scheduled solutions. Examples of this form of uncertainty include uncertainty in what routes are possible, uncertain task or constraint locations, and uncertain task or constraint requirements in terms of capabilities or types. To maintain explicitly scheduled coordination solutions in such domains will at a minimum require a repair mechanism to update schedules to accurately reflect changing world state information. Merely repairing solutions and continuing execution may not be sufficient to achieve high quality performance, however, as new information may significantly alter the coordination space. Thus for many domains reactive re-planning that goes beyond solution repair may be warranted, even for short horizon approaches.

Coping with high levels of uncertainty has many implications in terms of approach design. Reacting quickly to new information becomes a central concern approach design; delay in re-planning can harm performance, and even short horizon approaches will frequently require repair or re-planning. The benefits of long-term horizon become more limited in domains with high uncertainty. Explicit scheduling approaches that use a long horizon search the space assuming that the current world state represents a static domain problem instance. In domains with high uncertainty a static domain assumption becomes increasingly dubious. In such domains short horizon approaches that do more extensive reactive re-planning will generally outperform longer horizon, less reactive approaches. Thus in this section we focus on adapting short horizon approaches to cope with high uncertainty.

For domains with extremely high levels of uncertainty, maintaining explicit schedules may become impossible. Instead a more implicit notion of coordination should be adopted, potentially involving statistical modeling or expert knowledge of the environment. Such an approach could resemble a more sophisticated version of our independent tasks approach. We leave exploration of such an approach for future work.

In this section we focus on a version of the disaster response domain with uncertainty that undermines explicit scheduled coordination. In this version of disaster response we assume that task locations and the route map are known, but that the locations of debris piles are unknown. Agents only learn of intra-path constraints when the constraint sites are sensed by some agent. Explicitly scheduled solutions will generally be conceived with limited knowledge of constraint locations, and will need to be adapted in response to learning of new constraint locations. We implement two approaches to reactive re-planning for our short horizon, explicit scheduling approaches. The first method is a repair approach that minimally adjusts schedules to accommodate new constraints. The second approach is a more extensive reactive re-planning approach that considers new routes and assignments in response to new constraint information. In the rest of this section we describe these different approaches and evaluate the approaches on a simulated disaster response domain with uncertainty in constraint locations.

6.2.1 Coping with unknown intra-path constraint locations

In this discussion we assume that while fire locations and the overall shape of the road network are known *a priori* – determined by aircraft or satellite imagery, perhaps – debris pile locations cannot be determined until agents get within a certain distance of a pile - this is the sensing distance of the agents. We use a basic model of sensing where constraint locations within a certain radius – as defined by a parameter SENSE_DISTANCE – of any agent are discovered, and that once discovered locations are broadcast to all agents. While explicitly scheduled solutions can be determined through the area that has been sensed by the team, outside of that area it is likely that unknown intra-path constraints will prevent agents from reaching task locations as scheduled.

The independent tasks approach requires few alterations to cope with uncertainty in this form. Coordination is already implicitly scheduled, so there are no explicitly schedule interactions to repair. Fire trucks will continue to follow the shortest path to reach their assigned fire, waiting until constraints are addressed. Newly discovered constraint locations can simply be included in the next auction for bulldozer assignments, and will be addressed if they can be satisfied more quickly than other constraints.

Both the ATC and TA-IA approaches use explicitly scheduled coordination, and the coordination solution may incorrectly represent the world state if new constraints are discovered on agents' current routes. We do not alter the original strategies of the approaches - tasks are assigned and routes determined based on the currently known world state. If new constraints are discovered on intended routes, however, then we have two approaches for maintaining explicit scheduling. First, existing schedules are updated using a repair mechanism, where additional constraint-addressing resources are recruited to address newly discovered constraints. Once this recruitment has occurred and been scheduled fire trucks will continue along their previously determined routes. This approach maintains explicit scheduling, but does not account for the fact that extra delay may be incurred, reducing the solution quality. The second method we use does more extensive reactive re-planning, attempting to reason about the potential for altering the current coordination solution to improve performance. The re-planning approach offers greater reactivity to new information, potentially resulting in higher performance though at increased computational cost.

Schedule repair

Our repair approach seeks to do the minimum amount of work such that explicitly scheduled coordination is maintained when new intra-path constraint sites are discovered. We accomplish this through a similar auction framework to that used during the route search process. In our approach when new intra-path constraint sites are discovered by any agent the information is broadcast to all agents. Fire truck agents then attempt to determine if their current schedules are affected by this information - this will only be the case if an agent's current route takes it through the newly discovered intra-path constraint site. If affected will attempt to allocate the new intra-path constraints by holding constraint auctions. Bulldozer agents bid based on adding tasks to the end of their schedules, or if currently executing the Idle behavior based on empty schedules. Constraint satisfaction duties are awarded based on earliest completion time. Once the new constraints have been awarded and scheduled fire trucks will again have an accurate, explicitly scheduled solution given the currently known set of intra-path constraints. This process should be quite fast as agents do not attempt to alter their routes or other aspects of their schedules; only a single auction round needs to be held to allocate each newly discovered debris pile.

More extensive re-planning

Schedule repair should be computationally inexpensive and serves to maintain explicitly scheduled coordination when new constraint locations are discovered; it does not consider whether the presence of new constraints means that another coordination solution would be preferable. The presence of new intra-path constraints can significantly affect the relative quality of a coordination solution, and doing re-evaluation that goes beyond schedule repair can potentially improve performance. Our first attempt at more intensive reactive re-planning was to redo route search for agents whose routes were affected by the discovery of new constraints; this search should allow agents to consider whether a new route may offer a faster way to reach its currently assigned task. This approach did not consistently improve performance, as for many fires other potential routes to the fire would frequently also be blocked with previously unknown intra-path constraints.

We then attempted an even more extensive re-evaluation of the solution, considering whether a different allocation would improve performance after fire truck schedules are affected by newly discovered constraints. This approach combines schedule repair with re-assessment of the current allocation. The approach we take is to initially have fire trucks repair schedules as described above. When a new auction round for fire allocation is conducted, however, any truck that was affected by newly discovered debris will bid in the auction even though they have already been assigned a task. The process these agents use for bidding has two important differences from the standard bidding algorithm. The first major difference is that agents bid based on marginal average reward – the difference between the average reward the truck will receive for task offered in bidding and the average reward it expects to receive from its currently assigned task. Bidding marginal average reward accomplishes two things. First, agents will only be assigned a new task if its bid is positive - if it will receive a greater average reward from accomplishing the new task. If it cannot improve its reward based on addressing a new task it should continue to execute its current plan. Secondly, if an agent is competing against other agents in the auctions its bids will be at a disadvantage. Even if the agent which has already been assigned a new task could potentially gain greater average reward from accomplishing a new task, it will generally be superior in terms of overall solution quality to assign the task to an idle agent that can make slightly less.

The other main difference between standard tiered auctions and our re-planning approach concerns bulldozer responsibilities. Fire trucks will only bid if they were affected by newly discovered debris - if they were affected it must mean that one or more intra-path constraints must be satisfied to allow them to reach their goals. As schedules were accurately explicitly scheduled previously and corrected as part of the repair process some bulldozers must be assigned to clear debris piles along those routes. If the agent abandons its current assignment to address another task – a replacement task – it may be the case that bulldozers are assigned to clear debris piles which no longer lie on a route that some agent is currently planning to take to a fire. This will generally represent an inefficient solution, especially if the truck that has been assigned a new task must then wait longer for constraint-addressing assistance along their newly selected route. It can potentially improve performance to have bulldozers freed from the obligation to address constraints that are no longer explicitly required by some task-addressing agent.

In order to take this information into account during bidding bulldozers must be informed if some previously scheduled constraints need not be addressed if the new schedule is adopted. To enable this we include extra information in the auction call. When a task-addressing agent generates a new auction context it also calls a function that takes both the agent's current schedule and the new route schedule. This function iterates through the current schedule and checks to see if the allocated constraints are also in the new route. If not it adds them to a list, which is then attached to the auction calls for the intra-path constraints in the route. When bulldozers receive the auction call and the list of constraints C that are no longer required for the given context they first copy their current schedules into new context schedule S and then iterate through the context schedule, considering each assignment in turn. If a bulldozer determines that one of its current assignments is in the list C, and if the bulldozer further determines that no other agents besides the auctioneer requires the constraint to be addressed, then that constraint is deleted from the context schedule. The sub-auction process will yield accurate bids based on altering bulldozer commitments. If a task is awarded in lieu of a current obligation, the winning truck informs the auctioneer that its previously assigned task is again available and bulldozers adopt their associated context schedules, whether those schedules involve addressing new constraints or removing existing commitments.

To illustrate the benefits of reasoning about altering bulldozer commitments during the task replacement auction, consider Figure 6.3. In this example debris piles 1,2, and 4 were just discovered by the fire truck. Schedule repair yielded the schedules shown by the solid arrows in the figure. The fire truck is likely to be substantially delayed waiting for the bulldozer to address both constraints, and the path to the other fire is blocked by only a single debris pile. If the truck had to wait for the bulldozer to address constraints 3 and 4 before it addressed 1, then extinguishing the fire in the right of the image would remain the best solution. Using our method, however, the truck can inform the bulldozer that it need not address constraints 3 and 4 if the new route is adopted, with the end result being that an improved solution can be determined; the schedules that result are indicated by the dotted arrows.

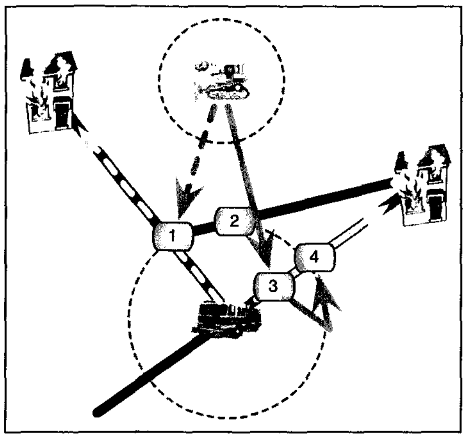


Figure 6.3: An example illustrating the task replacement auction process. The dotted grey circle is the extent of the sense distances for the agents. The solid arrows represent the current schedules and the dotted arrows the schedules used in task replacement auction bidding.

Experiments and evaluation

In this section we test in a version of the disaster response domain with uncertainty both in dynamic task issue and in constraint locations. We test with 5 different approaches - independent tasks, ATC with schedule repair only, TA-IA with schedule repair, TA-IA with schedule repair and reactive re-planning, and TA-TE with schedule repair. Five fire truck agents and 10 bulldozer are operating in the domain. 60 fires are issued at time zero, and a value of $\lambda = 1$ is set for the dynamic task issue rate. In these experiments we test on the same 10 domain instances with 200 debris pile intra-path constraints for 150 time cycles. We test the affects of limited perception by varying the parameter SENSE_DISTANCE, starting at a distance of 1, then moving to a distance of 2, and finally testing with no uncertainty in constraint locations to get a baseline of performance. Figure 6.4 shows a visualizer screen shot of simulation where the SENSE_DISTANCE is set to 1. As this parameter increases agents have more and more advanced notice of new constraints and, generally speaking, constraint locations will be learned more quickly.

Performance results are shown in Figure 6.2.1. We first look at the results at the infinite sense distance level, where the only form of uncertainty is that associated with dynamic task issue. In Chapter 5.2.2 we noted that the performance gap between TA-TE and TA-IA was narrower for domains with a relatively small number of constraint-addressing agents, and in Chapter 6.1.4 we showed that the gap between these two approaches is narrower for domains with dynamic task issue. This domain parametrization has both dynamic task issue and few bulldozers; the result is that the TA-TE and TA-IA approaches achieve roughly similar performance. The added reactivity of the shorter time horizon approach balances out the advantage of planning further into the future. We next observe that while performance of both TA-IA with repair and TA-IA with repair and

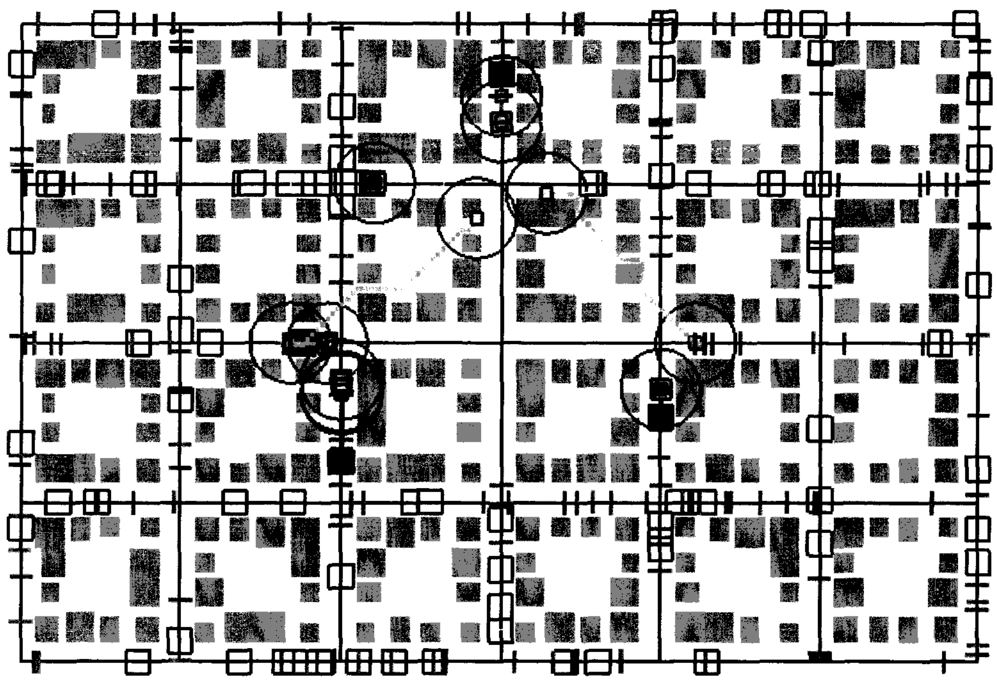
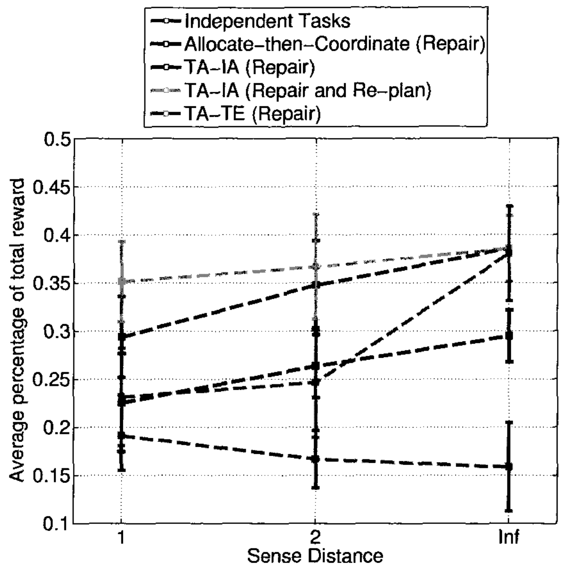


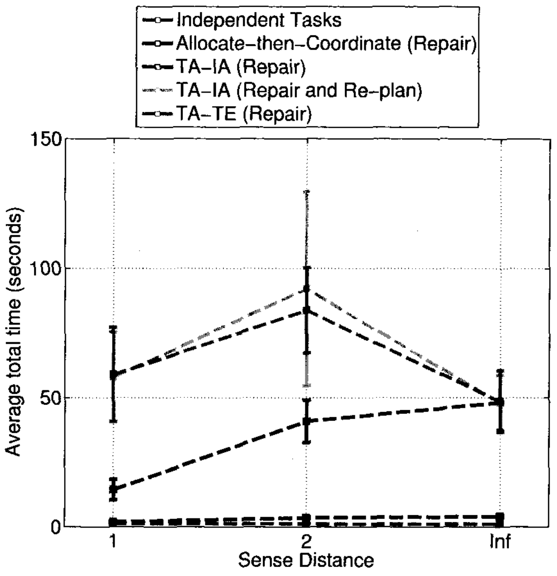
Figure 6.4: A screen shot from our simulation visualizer in a disaster response domain with uncertain constraint locations where the SENSE_DISTANCE parameter has been set to 1. The black circles around agents represent the sensing range; magenta lines parallel to the roads represent sensed constraints, and the blue lines are unknown constraints

re-planning decreases as the sense distance is lowered, the re-planning approach degrades much less steeply. At the 1 sense distance level the re-planning approach outperforms the repair approach by almost 20%. As the sense distance decreases the coordination solution accounts for increasingly few constraints during planning, making re-planning more useful.

The performance of the TA-TE approach decays much more steeply than the TA-IA with repair approach. The time-extended mechanism generates long time horizon plans based generally on incomplete world information. Without a mechanism for substantial re-planning, constraint-addressing agents quickly become saturated addressing constraints along agents' frequently long routes. Though the TA-TE approach gains some reactivity by using partial re-planning in response to dynamic task issue, the longer planning horizon becomes detrimental for domains with this level of uncertainty. A final interesting note in terms of performance is that the independent tasks approach actually improves as we decrease the sense distance. We postulate that the limited sense distance actually serves to focus constraint-addressing agents on the constraints that are discovered during the attempted execution of routes. Once discovered these constraints are more likely to be addressed, whereas more distant constraints may remain unknown and will not be addressed. This insight could potentially be used to develop a better heuristic for implicitly scheduled allocation.



(a) Average reward achieved by approaches as a proportion of an upper bound of available reward, where the bound is the summed reward of all issued tasks if they were addressed immediately. Standard deviations are shown as error bars.



(b) Average total time taken to produce coordination solutions. Trials were all run in a single thread and allowed 100% of a CPU on a quad-core Intel Xeon 3.8 GHz CPU.

Figure 6.5: Approach performance on a disaster domain with unknown constraint locations and limited sensing distances.

Figure 6.5(b) shows the computation used by the approaches. In the infinite sense horizon case, TA-TE and TA-IA use almost identical computation. Overall, the TA-IA approach is the most expensive - the added re-planning done in this approach comes at a substantial computational cost. Adding reactive re-planning to repair more than doubles the computation time at a sense distance of 2 and increases it by almost a factor of 4 at a sense distance of 1. TA-TE is almost as computationally expensive as the TA-IA with reactive re-planning approach, and performs less well than the ATC approach, which is orders of magnitude less computationally expensive. While an extensive reactive re-planning approach could potentially improve TA-TE performance, the benefits of long horizon planning are likely to be minimal in a domain with such a high level of uncertainty. We note that decreasing the sense distance does not necessarily reduce the computation requirement. For the TA-IA approach, the jump in computation time between the infinite and the sense distance of two can be explained by the extra time required for re-planning. For the TA-TE approach the NUM_DEBRIS_CONS bounding method is less effective when the full set of constraints is not known, causing the required computation to increase when there is limited sense distance.

6.3 Summary and discussion

In this section we examined the effects of moving beyond static domain instances to dynamic domain instances, where substantial new information is discovered during the course of execution. Many real world domains manifest some form of dynamism, and handling uncertainty is essential in such domains to achieving high quality coordination solution over the course of execution. We first discussed how we could potentially adapt our approaches to operation in domains with dynamic task issue, a form of uncertainty that does not necessarily require revising the explicitly scheduled coordination solution but where reactive re-planning can improve performance. Our short horizon approaches can operate in dynamic task domains essentially without alteration, as they already employ frequent re-planning. For our longer horizon approaches we described a partial re-planning approach, which seeks to minimally alter agent schedules in response to dynamic task issue. Our approach employs opportunistic path planning, and tasks are awarded to agents that can improve their schedules the most by opportunistically adding new tasks to their current schedules. We showed that this approach improved performance both in the TA-TE approach and the GA approach. The GA approach has no built-in provision for re-planning during execution, and we describe a method for periodic full optimization. This adds substantially to the computation cost of employing the approach. While the performance hierarchy established in the previous chapter remains the same, the performance gaps between the more reactive instantaneous allocation approaches and the time-extended approaches narrow to some extent.

We also examined handling a form of uncertainty that affects the accuracy of explicitly scheduled coordination solutions. We focused on a domain with unknown constraint locations which can only be sensed when agents move within a certain distance. In order to maintain correct explicitly scheduled coordination solutions approaches must at a minimum repair schedules by recruiting and scheduling constraint-addressing resources. In addition to repair there is also the potential to do more extensive reactive re-planning, as the quality of an intended allocation or route can be substantially altered when new constraints are discovered. We described a computationally inexpensive method to repair schedules where task-addressing agents hold auctions to recruit assistance if new constraints lie on agents' intended routes. We also described a more extensive reactive re-planning strategy where task-addressing agents will potentially be allocated new tasks if constraints are discovered on their intended routes. While more extensive re-planning is computationally costly, employing such re-planning improve TA-IA performance substantially over a repair-only approach, especially as the distance at which agents can sense constraints is reduced. We also showed that planning further into the future is actually a liability in domains with high uncertainty, as the TA-TE approach with repair achieves far lower solution quality than a TA-IA approach with repair.

We summarize strengths and weaknesses of the approaches in domains with different kinds of uncertainty in Figure 6.1. Introducing uncertainty into the domain significantly complicates our no-

Approach	Dynamic task issue	Short sensing ranges	RoboCup Rescue levels of uncertainty
Ind. Tasks	−	+	+
ATC	−		
TA-IA	+	++	−
TA-TE		−−	−−
GA	−	−−	−−

Table 6.1: General guidelines of the strengths and weaknesses of the approaches in dynamic instances with different characteristics.Strengths are denoted with the + symbol, and weaknesses with the − symbol, with particular strengths or weaknesses doubling these labels.

tions of which approaches to employ and where additional computation is most beneficial. In static domain instances the trade-offs were straight-forward: considering more of the coordination search space nets additional solution quality, though may require additional computation time. When uncertainty is considered the trade-offs becomes more complicated, as reacting to new information becomes a potentially important part of the solution quality equation. We have shown that we can improve the reactivity of longer horizon approaches by explicitly re-planning in response to new information. The question then becomes whether or not longer horizon approaches are justified in domains with uncertainty - is using extra computation to plan further into the future a worthwhile use of resources given a rapidly changing coordination space?

For dynamic task uncertainty our answer to this question is a qualified yes. For the market-based approaches while the performance gap between TA-IA and TA-TE narrows, a substantial performance gap still exists, and the TA-TE approach requires only moderately more time for coordination. The TA-TE approach benefits from the use of a computationally inexpensive partial re-planning approach. For the genetic algorithm the answer is more equivocal. While the GA with regular full re-planning cycles can outperform the TA-TE approach, doing so now requires 3 orders of magnitude more computation. The GA approach potentially wastes a good deal of computation time, planning far into the future when the shape of the coordination structure will likely substantially change before more than a small portion of the coordination solution can be executed. The use of the GA approach is only justified if computational resources are practically unlimited.

In the case of more substantial forms of uncertainty, such as unknown constraint locations, using long time horizon explicit scheduling approaches are almost certainly unjustified. While there is the potential that frequent re-planning could improve the performance of such approaches, planning further into the future based on an expectation that the world state is known represents largely wasted effort. Some domains manifest uncertainty even more drastic than unknown constraint locations. As the level of uncertainty increases the whole project of explicitly scheduling coordination becomes dubious. Each of our explicit scheduling techniques, during planning or re-planning, searches the coordination space assuming that the current world view represents the correct world view - essentially assuming that the current world view represents a static domain instance. For domains with an expectation of uncertainty this represents a poor assumption. In domains with an expectation that schedules do not reflect realistic assessments of time the line between explicit and implicitly scheduled coordination blur, as in neither approach can search in the coordination space rely on accurate evaluations of solution quality. One possible approach is to attempt to supplement the explicitly scheduled approaches with implicit reasoning. For instance, it may improve performance in a domain with unknown intra-path constraints to lower the reward estimate for tasks that require navigating through a good deal of unknown territory, as it is likely that intra-path constraints will delay the completion of the task. Alternatively, it may make sense to assign a bulldozer to follow a fire truck when it takes a route that moves it through unexplored space. Anticipating uncertainty may make the coordination solution more robust.

At some point, uncertainty may be so extreme as to functionally preclude explicit scheduling. As we described in Chapter 2.3.1, the RoboCup Rescue Simulation League is designed to test approaches in domains with high levels of uncertainty and hard constraints on the amount of time available for determining coordination solutions. Agents do not know the locations of intra-path constraints or tasks, and communication is unreliable. In a domain with this level of uncertainty attempting to plan with a long time horizon using explicitly scheduled coordination is unlikely to be a good use of computational resources. Approaches that have been successfully demonstrated in such domains place less of a focus on explicit coordination and more on reacting to new information and anticipating where certain kinds of agents will be needed.

Chapter 7 Beyond precedence

In the previous chapters we have focused on intra-path precedence constraints that exist in domains where some agents require terrainability assistance, detailing a number of different approaches to such domains as well as how to adapt these approaches to handle different kinds of uncertainty. To fully treat the topic of domains with intra-path constraints we also must consider other kinds of intra-path constraints and requirements that differ from terrainability assistance. In this chapter we first expand our scope to include disaster response variations that involve not just intra-path precedence constraints, but also varieties of simultaneity constraints: simultaneity requirements between constraint-addressing agents at precedence constraint sites, simultaneity between task-addressing and constraint-addressing agents at constraint sites, and simultaneity requirements between task-addressing agents at task sites. Adding simultaneity constraints increases the interrelations between agents' schedules, as now agents must be co-located at particular times to satisfy the constraints; accommodating this level of inter-relation significantly complicates the coordination problem. A mitigating factor in this added difficulty is that the spatial structure in domains with intra-path simultaneity constraints. Given a certain set of assumptions and minor alterations, our suite of approaches can still determine high quality solutions in domains with intra-path constraints of both simultaneity and precedence.

Disaster response and other domains with terrainability assistance requirements represent only one segment of potential domains with intra-path constraints. We are also interested in intra-path constraints in domains with collaborative resource acquisition, where constraint-addressing agents assist task-addressing agents in acquiring resources that allow domain tasks to be accomplished. In this chapter we also consider the problems of coordination in the on-order manufacturing domain, describing both the potential to apply approaches from our disaster response and identifying new problems for future work.

We first address simultaneity requirements between constraint-addressing agents, and then discuss domains with simultaneity requirements between task- and constraint-addressing agents. We next address the problem of coordinating agents when there are simultaneity requirements such that multiple task-addressing agents must simultaneously co-locate at task sites to address tasks. Finally, we consider extending our approaches to accommodate the requirements of on-order manufacturing.

7.1 Simultaneity constraints between constraint-addressing agents

The first form of requirements we will consider are domains where constraints of simultaneity exist between constraint-addressing agents at intra-path precedence constraint sites. In disaster response a domain with this characteristic would require that some number of bulldozers simultaneously co-locate and cooperate in order to clear debris piles. In such a domain the work of addressing constraints cannot begin until the required number of agents co-locate, and all agents are required to remain for a specified duration to address the constraint. Adding this kind of constraint significantly affects the problem of determining coordination solutions. In domains where there are exclusively intra-path constraints of precedence, multi-agent constraints potentially exist between the schedules of task-addressing and constraint-addressing agents, but no constraints exist between different constraint-addressing agent schedules. This allows explicitly scheduled coordination to be determined without considering the actions of constraint-addressing agents in tandem. Incorporating simultaneity requirements means that interdependence between constraint-addressing must be considered when making allocation and sequencing decisions.

Of particular concern is the fact that ignoring multi-agent constraints when determining sequences can potentially result in **deadlock**, a situation where two or more agents can make no further progress in executing their schedules due to constraints that will never be satisfied. Consider, for instance, a coordination solution where two agents have been tasked with addressing two constraints A and B, each of which require two agents. Some method has been used to determine sequences for the agents where one agent intends to address B and then A, and the other A then B. The agents move to the first constraint in their schedule and wait for the conditions to be met so that they can continue schedule execution. In this example the agents would delay at the constraint sites indefinitely, producing deadlocked behavior. In order to avoid deadlock and unnecessary delays we must produce constraint-addressing sequences that are largely synchronized, where the different agents assigned to the same constraint move to address that constraint at roughly the same time.

7.1.1 Adapting independent tasks

We first consider the problem of adapting the independent task approach to produce reasonable solutions in domains with simultaneity requirements at intra-path constraint sites. Adapting the independent task approach to cope with simultaneity constraints is relatively straightforward, as agents do not sequence constraints. The main required modification to the algorithm is that we now must consider how best to allocate single constraints that may require the simultaneous efforts of multiple agents. In our original formulation of the independent task approach described in Chapter 3.4 constraints are allocated on the basis of using bids to determine which constraints could be satisfied the most quickly. When constraints involve the simultaneous efforts of multiple agents, however, agents cannot in isolation determine at what time constraints will be satisfied; only when the arrival times of a number of agents are considered in concert can a constraint completion time be determined. While agents cannot independently determine their completion time for tasks, they can independently determine arrival times at tasks. By bidding the arrival times for tasks enough information is centralized at the site of the auctioneer so that assignments can be determined and schedules set.

Algorithm 10: Independent tasks approach for constraint allocation where constraints may involve simultaneity requirements

```
Input: T: List of unaddressed constraints
       A: List of constraint-addressing agents
Result: Single constraint assignments that respect simultaneity requirements for
        constraint-addressing agent
1  DeleteAgentAssignments(A)
2  foreach a in A do
3      agentBidList ← ComposeAgentBidList(a,T)
4      Add agentBidList to agentBidLists
5  Determine bidListsForConstraints from agentBidLists
6  while A is not empty do
7      foreach l in bidListsForConstraints do
8          if l.size < l.constraint.num then
               // l.constraint.num has the number of agents required in the
                  constraint. This condition occurs when there are not enough
                  unassigned agents to meet constraint requirements.
9              l.completionTime ← INFINITE_TIME
10         else
11             Sort the bids in l by increasing arrival times
12             l.startTime ← GetNthArrivalTime(l,l.constraint.num)
13             l.completionTime = l.startTime + l.constraint.duration
                  // l.constraint.duration has the time required for addressing the
                  task once the requisite number of agents have arrived at the
                  constraint
14     Sort bidListsForConstraints by increasing completion time
15     l ← bidListsForConstraints.front // bidListsForConstraints.front will have the
          task with the earliest completion time
16     if l.completionTime = INFINITE_TIME then
17         break // There are not enough unassigned agents for any constraint.
18     for i ← 1 to l.constraint.num do
19         b ← GetIthBid(l,i)
20         Assign b.agent to l.constraint
21         Remove b.agent from A
22         Delete all bids associated with b.agent from bidListsForConstraints
23     Remove l from bidListsForConstraints
```

The algorithm for the independent tasks approach to constraint allocation is shown in Algorithm 10. First note that instead of submitting bids for a single constraint, bulldozer agents now must submit a bid for each constraint. In order to avoid doing an expensive, precise path plan for each task, agents submit approximate arrival time bids based on Euclidean distance. Once arrival time lists are composed, the algorithm composes another structure containing the arrival time bids submitted for each constraint. Each individual constraint bid list is sorted; a completion time for the constraint can then be computed by first determining the start time for the task, which is the arrival time of the nth agent to arrive at the constraint, where n is the number of agents required to simultaneously address the constraint, and then adding the required duration to the start time. The bid lists for constraints can then be sorted in order of increasing completion times. The first entry in the sorted bid list will be the constraint with the earliest estimated completion time. The first n agents in the sorted bid list for the winning constraint are assigned to the constraint. All bids for those agents are removed from the bid lists for remaining constraints, as those agents have already been assigned tasks and should not factor into determining constraint start times. The algorithm will then consider which constraints can be addressed the earliest by the remaining unassigned agents. The algorithm terminates when all agents have been assigned constraints or all remaining constraints require more than the number of unassigned agents - these agents will remain untasked. When some agents in the domain become idle due to completed constraints we dissolve agent commitments and run the algorithm again; this is necessary to allow constraints to be addressed that may require more agents than just the agents that become idle at any given time.

7.1.2 Adapting explicitly scheduled market-based approaches

We now move on to considering how to accommodate simultaneity constraints between constraint-addressing agents in our explicitly scheduled market-based approaches - ATC and both tiered auction approaches. These approaches consider sequences of constraints, and avoiding deadlock becomes a concern. While the problem of searching the full space of constraint sequences becomes more difficult when simultaneity constraints are introduced, we do not consider arbitrary constraint sequences in these approaches. Constraints are allocated and sequenced using the earliest completion time heuristic, which allocates constraints in the order that they will be encountered on task-addressing agent routes to the constraint-addressing agents that can address them in the shortest amount of time. Our existing implementation of this heuristic, with a few minor alterations, can be used to quickly determine high quality constraint allocations and sequences even if constraints require simultaneity between constraint-addressing agents.

The modifications affect the route search process. When fire trucks hold auctions to assign and schedule debris piles, bulldozers bid arrival times at constraint sites instead of the times at which constraints will be completed. Arrival time bids are sorted by earliest arrival. Given arrival times the process for determining a completion time is equivalent to that used in Algorithm 10. The earliest arriving n agents are provisionally awarded the constraint in the context auction; as part of the award the winning agents are also told the scheduled completion time for the constraint. The bulldozers can then update their context schedules with the completion time, as the completion time must be accounted for in future bids for constraints associated with the context. This method ensures that no deadlock occurs, and that bulldozer sequences and assignments will allow the truck to reach its destination fire or fires quickly. Beyond adjusting the idle behavior to resemble the independent task approach, our market-based explicit scheduling approaches need not be altered in any other way to produce high quality solutions in domains with simultaneity requirements at the sites of intra-path constraints.

7.1.3 Adapting genetic algorithms

While the market-based approaches need not consider arbitrary sequences of constraints, such sequences do factor into full coordination space search. For this reason the problem of determining high quality sequences for intra-path constraints given simultaneity requirements is most difficult in the genetic algorithm approach. Some alterations are relatively straight-forward. In terms of genome encoding, genomes must be adjusted to reflect the fact that more than one agent may be required to satisfy particular constraints. We do this by creating synthetic IDs for those intra-path constraints that require more than one agent. If a constraint requires n agents, we create n IDs associated with that debris - each of those ids represents an assignment to be one of the agents that addresses the associated intra-path constraint. In large part these synthetic IDs can be treated like any other assignment, with one significant exception.

In our GA approach developed for domains where constraints require single bulldozers we took care in constraint design that constraint assignments were unique, though after initialization this was only really a concern during bulldozer sequence crossover; all the mutation operators only act on a single genome, while in crossover different genomes are combined. When considering synthetic IDs, however, we need to make sure not only that assignments of IDs are unique but that agents are not assigned two different synthetic IDs associated with the same intra-path constraint. If an agent was assigned two synthetic constraints then the constraint would never be satisfied, as the required number of agents will never be simultaneously present at the constraint site. In order to prevent this we supplement the crossover algorithm to prevent duplicate synthetic ID assignments. We also need to add reasoning to prevent synthetic ID duplication to the mutation operators InterSwap and Reassign, the operators that alter assignments. In both of these operators we test whether or not an assignment alteration violates the exclusivity of synthetic ID assignments, retrying the operator for a specified number of times if duplication occurs.

Using the methods just described we can generate the full space of potential coordination solutions to intra-path assignments in domains with simultaneity requirements. However, the genetic algorithm may have difficulty searching this space using purely random operators. Aside from the inclusion of the SmartSwap operator the genetic algorithm approach described in Chapter 5 is largely non-heuristic. This approach produces good results for domains where schedules are significantly decoupled, but simultaneity constraints cause bulldozer schedules to be coupled. Non-heuristic approaches are likely to have serious difficulty coping with a coordination space where many agent sequences, when considered in tandem, result in deadlock or significant delay. While the algorithm can still potentially determine high quality solutions non-heuristically it becomes increasingly difficult to do so, as many solutions will be poor.

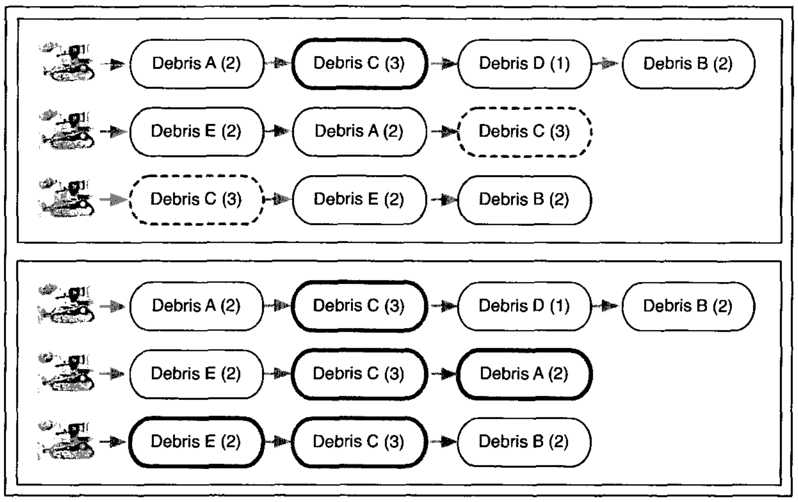


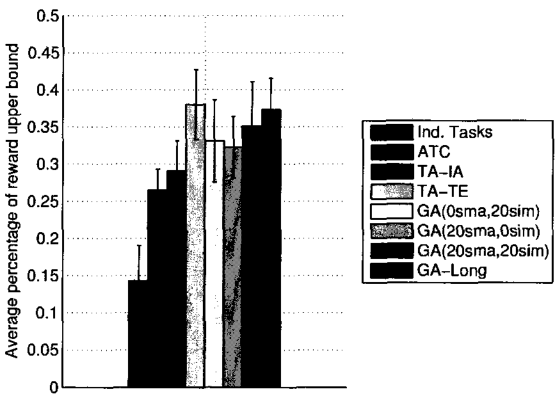
Figure 7.1: Example of the SimulSwap mutation operator, where the constraints in bulldozer sequences are represented as ovals, with the number of agents required by the simultaneity constraint in parentheses. In the top section of the figure Debris Pile C – with the solid red outline – in the top bulldozer's schedule is selected for the swap, with the other entries for Debris Piles C in the other sequences shown with dotted red outlines. The result of the SimulSwap is shown in the bottom figure, where the constraints swapped with Debris Pile C are shown in blue.

SmartSwaps can become even more beneficial in domains with simultaneity constraints. SmartSwaps tend to move constraints that are needed by fire trucks earlier in bulldozers' schedules. If several bulldozers are assigned synthetic IDs associated with a constraint that is required early in a fire truck's schedule and SmartSwaps cause them to advance that constraint in their sequences then a well-coordinated solution is more likely. In order to supplement this heuristic we developed another heuristic we call SimulSwap that specifically attempts to align the positions of the synthetic IDs associated with the same constraint in different bulldozers' schedules. The operation of the SimulSwap operator is shown in Figure 7.1. The SimulSwap mutation operator selects a single synthetic ID in a single bulldozers sequence is selected at random, with associated ordered position p – in the figure the selected constraint C is in position 2 in the sequence, but in positions 3 and 1 in the other sequences. The operator determines which other sequences contain the other related synthetic IDs. The mutation is to change the positions of the other synthetic IDs to match p, either precisely or at the end of the sequence if p is greater than the length of the other agents' sequences. The results of a SimulSwap are shown in the bottom of Figure 7.1, where the constraint occupying position 2 in each of the other schedules is swapped with the positions containing synthetic IDs associated with Debris Pile C. The effects of the SimulSwap may not be entirely positive - in the figure, the synchronization associated with constraint A is degraded by the swap. Despite this fact, SimulSwaps can potentially improve synchronization and lead to improved performance. Combining SmartSwaps and SimulSwaps is especially powerful, as SmartSwaps will tend to move needed tasks earlier in agents' schedules, and SimulSwaps will tend to align their positions.

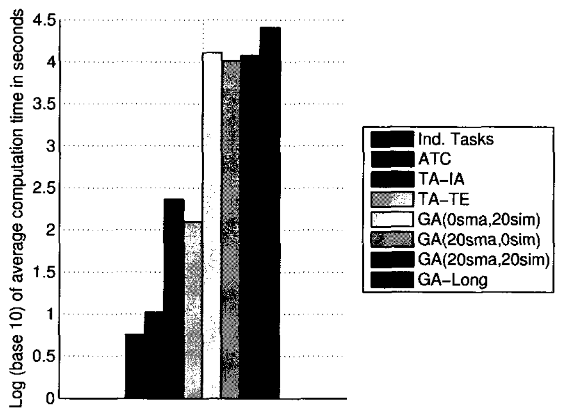
7.1.4 Experiments and evaluation

In this section we test the performance of different approaches in static disaster response domain instances with simultaneity constraints between constraint addressing agents at constraint sites. We test with 200 debris pile constraints, with a 40% chance that a debris constraint will require only a single agent, a 40% chance that two agents will be required, and a 20% chance that 3 agents will be required. There are three fire trucks and 21 bulldozers in the team. Domain parameters are otherwise identical to those described in Chapter 3.6.

We test with our standard suite of market-based techniques, which use the modified constraint assignment algorithm either for all constraint allocation in the case of the independent tasks approach or for bulldozer idle behavior in the other constraints. We test with four different parametrizations of the GA approach. The first three all use population sizes of 20,000 and optimize for 400 generations. The first of these approaches GA(20sma,0sim) - uses 20 smart swaps and no SimulSwaps per genome per generation; the second – GA(0sma,20sim) – uses 20 SimulSwaps and no smart swaps; the third – GA(20sma,20sim) – uses 20 of each. The final GA parametrization – GA-Long – uses 10 each of smart swaps and SimulSwaps, and runs with a population of 20,000 for 800 generations.



(a) Average reward achieved by approaches as a proportion of an upper bound of available reward, where the bound is the summed reward of all issued tasks if they were addressed immediately. Standard deviations are shown as error bars.



(b) Average total time taken to produce coordination solutions. Trials were all run in a single thread and allowed 100% of a CPU on a quad-core Intel Xeon 3.8 GHz CPU.

Figure 7.2: Approach performance in a disaster response domain with simultaneity requirements between constraint addressing agents at some constraint locations.

Performance results are shown in Figure 7.2(a). The performance hierarchy among market-based approaches remains the same. Among the genetic algorithm approaches, using 20 SimulSwaps exclusively over 20 smart swaps improves average performance slightly, though the improvements are well within the margin of error. Using both SmartSwaps and SimulSwaps improves performance even more. Running for more optimization iterations improves performance. In this case, however, none of the GA approaches outperforms the TA-TE approach. Even with the development of new heuristic operators, searching the space of sequences becomes considerably more of a challenge in domains with simultaneity constraints. There are many potential ways to improve the genetic algorithm performance. We could use a larger population size, run for even more generations, or design more powerful heuristic operators. In any case, adding simultaneity constraints seems to make the problem harder for full coordination search, whereas the more heuristic market-based approaches can accommodate the new constraints easily.

Required computation time is shown in Figure 7.2(b). Note that the Y-axis is the base-10 log of the total computation time. We first note that the cost of the minimal time approaches has increased substantially - the alterations to the structure of the independent task auction to accommodate simultaneity make it more computationally expensive. The TA-TE approach is somewhat less computationally costly than the TA-IA approach in this domain setting. The most computationally expensive GA approach, which still does not equal the reward of the TA-TE approach, takes 2.5 orders of magnitude more computation.

7.2 Handling simultaneity requirements for terrainability assistance

Another potential variation in terms of intra-path constraints also involves simultaneity at intra-path constraints, but in this case between task- and constraint-addressing agents. In these constraints task- and constraint-addressing must simultaneously co-locate at constraint sites in order for terrainability assistance to be rendered. This form of constraint can model situations where constraint-addressing agents must physically transport task-addressing agents to allow it to pass a constraint location. For instance, in the disaster response domain bulldozer agents may lack the capability to clear debris to make areas terrainable by fire trucks, but may have the capability to drag or carry fire trucks through the piles. While this form of constraint will cause task- and constraint-addressing agents schedules to become substantially more inter-connected, from the perspective of determining coordination solutions it can actually make the problem easier, as it can radically reduce the size of the coordination space.

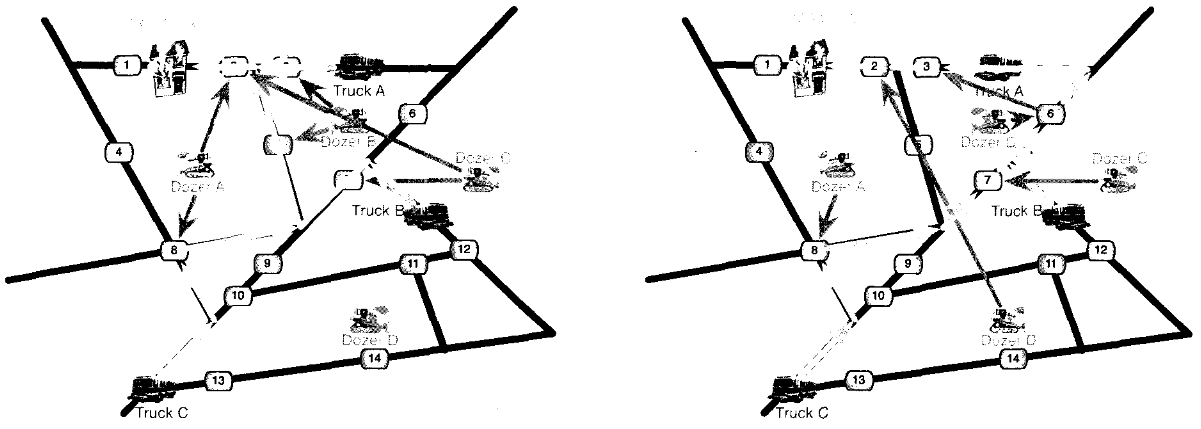
To illustrate this point, consider an instance where bulldozers can move at the same speed as fire trucks. In this instance, a fire truck is considering a route where there are three intra-path constraints with simultaneity requirements between trucks and bulldozers. Suppose that a bulldozer has been assigned to the first constraint allocated on the route. For the remaining constraints on the route there is no benefit to employing any other constraint-addressing agents. Given that the bulldozer can move as quickly as the fire truck, once the first constraint has been satisfied the bulldozer can simply follow the fire truck to the next debris site location and satisfy that constraint as well; unlike in the precedence intra-path constraint case there is no benefit to assigning multiple bulldozers to address constraints along a single route. In fact, if there are as many bulldozers as fire trucks the best approach is just to assign each bulldozers to follow a fire truck at all times. Once each truck has a co-located bulldozer, determining the best route to a fire can be done using single-agent planning.

Somewhat more complex cases with this form of constraint involve domains where bulldozers move more slowly than fire trucks or where variable numbers of constraint-addressing agents are required at task sites. If bulldozers move more slowly than fire trucks, the earliest completion time heuristic could be employed to determine reasonable solutions. If there are also constraints between constraint-addressing agents, the techniques we describe in the previous section could be used. In either case, our existing work should be sufficient to address domains with this form of simultaneity constraint.

7.3 Handling simultaneity at task sites

Adjusting our approaches to handle simultaneity between task-addressing agents at task sites is potentially more difficult than handling constraint simultaneity. In constraint simultaneity cases we can use the earliest completion time heuristic to largely mitigate the difficulties of sequencing; similar intuitions also contributed to our heuristic operator design for the GA. Our methods serve to assign the requisite number of agents to constraints; even if the constraints on the route call for different numbers of constraint-addressing agents the earliest completion time heuristic can produce a high quality allocation and sequencing of constraints to agents.

The general problem of determining the set of agents that should be involved in addressing a task that requires the efforts of multiple agents is a difficult problem known as coalition formation. As we described in Chapter 2.3.1, most coalition formation algorithms have focused on instantaneous allocation, where the concern is how to form coalitions across a set of agents given that each coalition will only be assigned a single task. This is a difficult problem in its own right, as an approach must consider what coalitions to form in addition to what tasks should be assigned to those coalitions. A standard assumption in the coalition formation literature is that the fitness associated with a given coalition is easy to evaluate - to determine coalitions an approach may need to search in an exponential space of possible coalitions and task assignments, but assessing the fitness of a given assignment to a given coalition is assumed to be a straight-forward computation. For constraint-addressing agents in domains with intra-path constraints this is a reasonable assumption; given a set of bulldozer agents, for instance, we can compute the completion time for the task by using single-agent planning to produce arrival and completion times as described in the previous two sections. For task-addressing agents, however, determining the fitness of a given coalition may be a difficult problem.



(a) Potential results if each agent independently conducted route search to reach the target fire W

(b) A better multi-agent route set and bulldozer assignments to allow fire W to be addressed as quickly as possible given full multi-agent path problem

Figure 7.3: Example illustrating the potential improvement in performance from considering the full multi-agent path plan for reaching a task with simultaneity requirements

Consider the example shown in Figure 7.3. We suppose in this example that we are trying to assess the value of assigning the coalition of three fire trucks to the fire W. In order to assign a value for the coalition it must be determined at what time agents will reach and then complete the task given the simultaneity requirements. To predict this completion time we must determine an explicitly scheduled coordination solution involving route search and allocation and sequencing of constraints to constraint-addressing agents. If we employed our route search algorithms for each agent independently assuming unfettered access to constraint-addressing resources, we might end up with the schedules shown in 7.3(a). Making an estimate based on whatever arrival times the fire trucks independently produce would be overly optimistic, as those estimates are based on conflicting bulldozer schedules. Even if we designed a method to de-conflict bulldozer schedules given fire truck routes, the route search process would not account for the fact that several fire trucks were simultaneously trying to reach a fire. Given this fact, Figure 7.3(b) shows a better selection of routes as well as constraint allocation and sequences given the full multi-agent path planning problem. This set of routes has a number of interesting features. First, the schedule results in none of the agents reaching the fire particularly quickly. Truck A will need to wait for Bulldozer D to travel a long distance to address Constraint 2, and for Bulldozer B to address constraint 6 before it addresses Constraint 3. Having Truck A reach the fire quickly has no utility, so it is better to allocate constraint-addressing resources to speed the passage of the other agents.

Second, the agents all eventually end up following the same route - once the agents co-locate in domains with intra-path precedence constraints the problem of route search becomes no harder than in the single-agent case.

The problem then becomes how to engineer an approach that is capable of producing this coordination solution while maintaining tractability. Even for instantaneous allocation of task-addressing agents to tasks, producing estimates of coalition quality involves considering task-addressing agents' routes in tandem. The heuristic methods we have employed thus far exploit schedule independence to achieve tractability, but for multi-agent path planning the problem of determining routes becomes highly interdependent. The heuristics we have developed in previous chapters do not readily extend to the multi-agent route planning problem, as even the more computationally intensive tiered auction approaches exploit the partial independence of agent schedules to make route search tractable. Our goal of determining high quality solutions quickly may be infeasible if we have to search in the multi-agent path planning space for determining the value of each potential single-task assignment to each potential multi-agent coalition. The problem becomes even more difficult when considering a time-extended coordination approach. In a time-extended approach the coordination approach must consider not only how to form coalitions to address individual tasks, but how the **coalition structure** – the set of coalitions formed at particular times – should change over time in order to maximize performance. Even for domains without intra-path precedence constraints the problem of adjusting coalition structure over time to maximize performance is largely unaddressed, though some work addresses this problem for multi-agent routing (Smith & Bullo, 2009) (Zheng & Koenig, 2008).

The general problem of tractably determining high-quality, explicitly scheduled time-extended coordination solutions in domains with intra-path precedence constraints and simultaneity requirements at task sites is something we leave for future work. However, given a few limiting assumptions about domain properties we can potentially extend our previously presented approaches to domains with task simultaneity. One primary insight that we exploit is that in domains with intra-path precedence constraints, once coalitions have co-located the problem of coordination can become significantly easier. Suppose that a coordination solution has been determined that results in three fire trucks co-locating at the site of a fire. Once that fire is extinguished, suppose that there is another fire that requires three agents at some other location in the domain environment. We showed in Figures 7.3(a) and 7.3(b) that it is a difficult problem to determine how three fire trucks at different locations should determine how to address a fire; the problem of getting three co-located fire trucks to another fire that requires three or fewer fire trucks is functionally equivalent to getting a single fire truck to a fire. Once agents are co-located – a process we call **rendezvous** – they can be treated as a single agent during planning with the capability to handle tasks that require a number of agents less than or equal to the coalition size. We can then use any of our approaches to determine coordination solutions for the set of co-located coalitions. A given coalition may not have the necessary size to address all domain tasks but reasoning about which tasks can and cannot be addressed can be easily incorporated into the auction process.

One approach that exploits this insight is to use a method to determine coalition structure, develop a method for rendezvous, and then treat the formed coalitions as single agents for planning purposes. While the problem of how best to achieve task-addressing agent co-location may be difficult in general, a strength of the above approach is that rendezvous need only occur once per coalition, as once coalitions are formed they remain co-located. Given that rendezvous need only occur once, it actually plays a small role in approach quality, and even simple, highly heuristic methods will be sufficient to achieve reasonable solutions. This approach formulation will work well in domains where it is straight-forward to determine a good coalition structure, and where there is limited benefit to changing the coalition structure over time. Take, for instance, a scenario where all tasks require two agents. In this domain the coalition structure that should be adopted is clear, and once agents are co-located there is not a good reason to alter coalition structure. While it may improve performance slightly to consider which agents should pair, and how the pairs should rendezvous, using computationally inexpensive heuristics to form coalitions will generally be sufficient.

In other domains the best coalition structure may be unclear, or there may be benefit to changing the coalition structure over time. For an example of when changing the coalition structure may improve performance, consider a domain where there are six task-addressing agents, and tasks require either one, two, or three agents. One choice of coalition structure would be to form two coalitions of three agents each. In this case, each of the two coalitions could address all domain tasks, and we could employ our approach methods once agents were co-located. However, when addressing tasks that only required one or two agents these coalitions would be larger than necessary, and fewer tasks may be addressed than if a different coalition structure was adopted. Another choice would be to form one coalition of three agents, one of two agents, and a single agent coalition, and have coalitions address only those tasks for which they had the requisite number of agents. This may represent a good coalition structure, but if it turns out that most tasks require three agents, then half the agents in the environment would be underutilized. A final possibility is that no fixed coordination structure can achieve maximal performance in a given domain. In our previous example, the best coalition structure may be to have two coalitions of two agents and two coalitions of one agents that can potentially combine to form three agent coalitions to address tasks that require three agents.

Altering coalition structure offers two significant challenges in terms of approaches to coordination, however. First, if the coalition structure is altered frequently then determining which agents belong in a coalition and how that coalition should achieve rendezvous becomes an increasing important component of the solution. Using computationally inexpensive methods for planning these difficult multi-agent path planning problems may seriously impair performance if they need to be frequently invoked. Second, changing coalition structure may come at a significant cost in terms of delaying task completion times. If it is determined that two coalitions should merge, for instance, and the two coalitions are far away from each other, then not only must a method for rendezvous be determined but agents may need to neglect other tasks while achieving rendezvous. An approach to evaluating changes in coalition structure must reason not only about the benefits of altering coalition structure, but the methods by which those changes can be effected and the cost in terms of delay for other domain objectives. Our greedy, heuristic approaches are poorly suited to evaluating the long-term effects of changing coalition structure, as given a short-term horizon it will almost certainly be better to maintain the current coalitions rather than delaying task completion to change the coalition structure. In domains where there are many different capabilities required to address tasks and the capabilities are spread between a number of different kinds of agents, assuming that coalition structure need not change will lead to very inefficient performance; we leave addressing such domains as future work.

Determining the best coalition structure for time-extended allocation, and altering the existing coalition structure to maximize performance are substantial research problems which we will not fully address in this thesis. Instead we will focus on computationally inexpensive methods to achieve co-located coalitions in domains where the coalition structure is clear and need not be altered over time. In the rest of this section we present two inexpensive approaches to rendezvous and illustrate that we can accomplish time-extended allocation using rendezvous and time-extended tiered auctions for co-located coalitions.

7.3.1 Achieving co-located coalitions

In this section we assume that all tasks require the efforts of two agents, and that an even number of agents are operating in the environment. Given the coalition structure of pairs of agents our goal then becomes to produce co-located coalitions in a manner that will maximize performance. There are two questions that an approach must determine - which agents should form pairs and where the agents should co-locate. We focus initially on the second question of how to achieve rendezvous between two task-addressing agents.

We suppose that two agents have been selected to form a coalition and consider the problem of rendezvous for the two agents. The general approach we took was to attempt to determine a rendezvous point, a point at which co-location will occur. Once the point has been determined we can set schedules for each agent that allow them to quickly reach the target point. We considered two potential computationally inexpensive approaches to this problem. First, we consider an approach where the agents rendezvous at a task location. The intuition behind the approach is that by having agents rendezvous at a task location once they reach the task they can address it, after which co-located planning will take over. The potential downside of this approach is that if the nearest tasks are far away it may take a good deal of time to achieve co-location. Our other approach to rendezvous tries to minimize the amount of time it takes to achieve co-location; in this method while agents may not be able to immediately address a task once they have achieved rendezvous, by achieving co-location more quickly more time can be spent in co-located planning mode using one of our approaches. A more computationally expensive method could potentially attempt to determine a good rendezvous point given the future assignments of the coalition, but we believe that the benefit from a slight improvement in selecting rendezvous points is unlikely to warrant using a substantially more computationally expensive approach.

In the first method, where a task is selected as the rendezvous point, each available task is considered in turn. The goal of the search is to find the task that the two agents can satisfy the most quickly. We determine this task using a heuristic based on shortest path distance without considering constraints. Each agent plans a shortest path to each task, and the task that can be addressed the most quickly given the simultaneity constraint is chosen; this will be the task where the second agent to arrive arrives the earliest, as the arrival time of the first agent is unimportant in determining when the task can be successfully completed. We could potentially improve performance somewhat by using individual agent route search to determine more accurate estimates of task completion times, or even consider the full multi-agent route planning problem for each task, but each of these approaches would substantially increase computational costs. The selected task becomes the rendezvous point. In cases where tasks may have greatly differing values it may be important to factor task value into this process instead of only factoring in which task can be addressed the most quickly. The process could be easily used to determine a rendezvous point for coalitions consisting of any number of agents.

Our second method for determining rendezvous points tries to select the location that will allow the agents to co-locate as quickly as possible. In this method we use a heuristic based on determining a midpoint between the two agents independent of constraint considerations. In this method we generate a shortest path from one agent to the other, and set a rendezvous location at the midpoint of this path. A more computationally expensive method could potentially factor in constraints to the midpoint calculation, which would help in estimating how quickly each agent will actually reach the midpoint location. This method would be more difficult to adapt to arbitrary numbers of agents.

Whichever method has been used to seleect the rendezvous point, we need to determine agent schedules that will allow them to reach the rendezvous point quickly. Instead of using a costly multi-agent route planning approach we simply do conventional single agent route search in a round-robin style. One agent determines a route via route search and makes the necessary constraint assignments, and the other agent then plans a route given the existing bulldozer schedules. If the rendezvous point is an unassigned fire, our approach fixes the task completion time in both schedules after the second agent has determined a schedule, as the completion time is a product of both agent arrival times. Once the agents have exhausted their schedules they will have achieved co-location and can be coordinated using any of our suite of approaches.

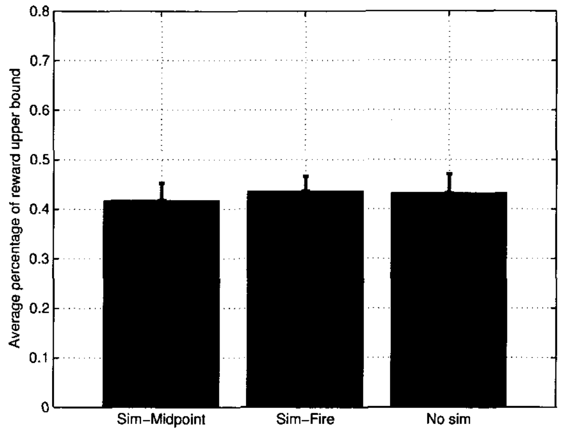
The previous methods allow us to generate schedules that will form and co-locate agents into a coalition, but we now need to consider how to determine which agents should be formed into those coalitions. In order to maximize performance we should consider this question in concert with determining rendezvous points and agent schedules: given a set of agents and constraints we would determine how best to form given coalitions, including which agents should form coalitions and how to achieve co-location in those coalitions. Maximizing performance to this problem is likely to be computationally expensive, and as co-location needs to occur only once for each pair, performance gains will likely be small. We adopt an approach of randomly selecting agents to form each coalition.

7.3.2 Experiments and evaluation

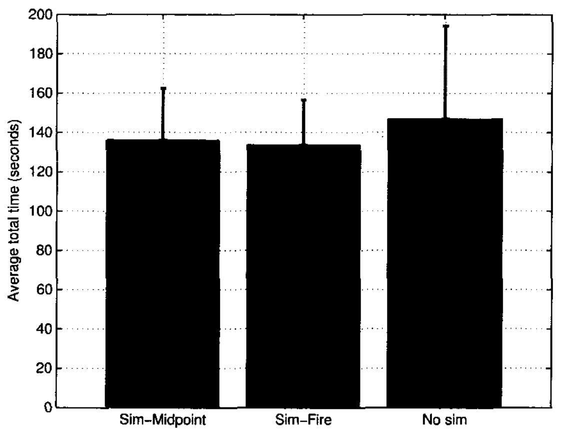
In this section we evaluate approaches within the context of determining coordination solutions in domains with simultaneity requirements at task sites and intra-path precedence constraints, given the assumption that the proper coalition structure is clear. In this case we suppose that six fire truck agents are operating in a domain where all tasks require the simultaneous efforts of two task addressing agents. The main goal is to show that we can successfully achieve rendezvous and achieve roughly similar performance to the performance of three fire trucks in a domain where tasks only require the efforts of a single agent. All tests are conducted with the TA-TE approach. For the simultaneous experiments we use two different approaches to rendezvous - fire rendezvous (Sim-Fire) and midpoint rendezvous (Sim-Midpoint). To test the equivalence to a domain without simultaneity requirements we alter the domain instances to include only three of the fire trucks included in the simultaneity domains, and change the fire requirements at tasks to single agent tasks (No sim). The domain instances are otherwise unchanged.

Performance results are shown in Figure 7.4(a). The fire rendezvous approach proves to perform slightly better than the midpoint approach. While midpoint may allow for faster rendezvous, the fire approach can potentially distribute coalitions into the environment more widely. The performance of the Sim-Fire approach is virtually identical to that in the domain instances with no simultaneity, showing that at least in these domain instances, where agents start relatively close together, even highly heuristic approaches to determining and forming coalitions do not substantially inhibit performance.

Computation times are shown in Figure 7.4(b). The rendezvous approaches actually use slightly less time on average than regular TA-TE. A large portion of the time taken in the standard TA-TE approach is directed towards the initial allocation at time 0, where all agents are idle and thus are competitively bidding for tasks. Agents using our rendezvous approach will not necessarily engage in this competitive bidding, as all the coalitions will reach their assigned rendezvous points and begin bidding for tasks at different times.



(a) Average reward achieved by approaches as a proportion of an upper bound of available reward, where the bound is the summed reward of all issued tasks if they were addressed immediately. Standard deviations are shown as error bars.



(b) Average total time taken to produce coordination solutions. Trials were all run in a single thread and allowed 100% of a CPU on a quad-core Intel Xeon 3.8 GHz CPU.

Figure 7.4: Approach performance in a disaster response domain with simultaneity requirements at task sites in addition to intra-path precedence constraints

7.4 Collaborative resource acquisition domains

To this point in the thesis we have primarily focused on our approaches situated within the disaster response domain. The work presented thus far should be directly applicable to other terrainability assistance domains, but the extension to collaborative resources acquisition domains requires a closer examination. In this chapter we discuss how domains with intra-path constraints involving resource acquisition are similar to and different from terrainability assistance domains, paying particular attention to the potential to adapt our approaches to achieve high quality coordination solutions in domains of this nature. We particularly focus on the on-order manufacturing domain discussed in Chapter 1.1, where agents must collaborate to manufacture objects for shipping.

We will initially focus on static domain instances of on-order manufacturing and exclusively on intra-path constraints of precedence. Later in the section we will discuss how the approaches can be adapted to cope with uncertainty and handle additional constraint types. In each static domain instance there are a known set of orders to be filled, each with a reward value and a decay function conditioned on time. There are some number of platform agents, which have a capacity constraint associated with the number of objects they can hold for manufacture. There are additionally tower agents associated with each tower, which have a capacity constraint associated with the number of components they can fetch from tower inventory in a single trip. Finally, there is a capacity constraint on the number of components that can be held in each tower staging area. The components must be placed in the staging area before they can be attached to the objects being manufactured.

As is the case in disaster response, a number of factors govern the particulars of the scenario, and affect how well different coordination approach methodologies will perform and which is most appropriate. In terms of the problem specification, it may be the case that there is only a single recipe for constructing objects, or there may be many potential methods. The larger the space of potential recipes, the larger the space of potential routes for item manufacture. Additional important factors include the number of components in inventory and the distribution of the components among towers. If, for instance, it is the case that there are only a few towers containing a particular component that is required for the manufacture of many different parts, then those locations could become bottlenecks.

In the rest of this section we will discuss each of suite of approaches, discussing the potentials for adapting the approach to achieve high quality solutions in resource acquisition domains.

7.4.1 Independent tasks

The applicability of independent tasks for a given domain largely depends on a designer's ability to determine a reasonable strategy for determining implicitly scheduled coordination. The main challenge in terms of design is determining on what basis coordination decisions will be made. For platform agents, this requires some method to determine the order assignments to agents, to determine a recipe for manufacture given assignments, and to determine routes given recipe selection. For tower agents, some method must be established to determine what tower agents will fetch and when.

For platform agents, order assignment could involve consideration of the importance of tasks, or be based on the length of associated recipes and the availability of parts. Once assigned an item to manufacture, the agent should move to try to obtain the first required part, and once the part has been obtained should move to obtain the next part and so on. Good strategies for obtaining parts will in large part depend on the strategy used for tower agent planning. In the disaster response domain, designing an independent task approach was straight-forward as there was only a single method by which bulldozer agents could satisfy constraints, and a fixed list of constraints to be addressed. In the on-order manufacturing domain, however, the list of intra-path constraints to be satisfied is contingent on the assignments and plans or platform agents, and there may be a large list of potential components that could be required for manufacture. This means that even an implicitly scheduled approach will need to carefully consider what components should be fetched and by which towers. One method could involve including domain information about the most needed parts - tower agents could tend to probabilistically fetch parts that were more generally needed, for instance. Another approach would be to attempt to ensure that some stock of each part was available at some location in the environment. In this method tower agents could potentially be assigned responsibilities to keep a certain stock of a small set of parts on hand, replenishing the supply when parts were used by platform agents. There could also be a system where platform agents request particular parts, and assignments to fetch those parts are determined using a metric such as which agent can fetch the parts the most quickly.

The performance of such an approach will depend on a number of different factors. In a domain with relatively few kinds of components in inventory, and where there are many different towers which can supply each part this approach should perform well - these versions of the domain are relatively straight-forward, akin to disaster response domains with few intra-path constraints. This approach may result in relatively poor performance in domains with large component sets, or where there are bottlenecks in performance - these domains are more akin to domains with high intra-path density in disaster response.

7.4.2 Allocate-then-coordinate

Depending on the particular features of domain instances an independent task approach may yield reasonable performance without using explicitly scheduled coordination. For more difficult domains, however, explicitly scheduling coordination should improve performance, though will likely involve additional computational cost. An explicitly scheduled solution will have tower agents fetching certain components and bringing them to their staging areas by certain times, where they will be picked up by particular platform agents. An ATC approach to the problem will use some method for task allocation to task-addressing agents that does not involve determining explicitly scheduled coordination. Once assigned an order, the platform agent will conduct search in the space of potential recipes and routes that will allow for the acquisition of components and the assembly of the objects.

The process of determining what recipe to take and allocating and scheduling constraints should be similar to the route search process in terrainability assistance domains. For a given recipe, the agent will have a space of potential routes that include the set of towers at which the first component in the recipe can be acquired, the set of towers with the second component in inventory, and so on for each component in the recipe. The earliest completion time heuristic should be useful in determining at which locations certain parts should be acquired, though the speed with which a component can be acquired will be a product of both the speed with which the component can be obtained and placed in the staging area by the tower agent as well as the required travel time to reach the tower by the platform agent. Bounding techniques will be required to search the space of potential recipes and routes quickly, and heuristics may be beneficial in searching the space of potential routes.

One substantial difference between on-order manufacturing and disaster response is that there is location choice in on-order manufacturing - constraints can potentially be addressed at a variety of different locations. The presence of location choice offers the opportunity for heuristics to be developed for use during the route search process. Consider a case where several towers with associated staging areas lie near each other. If it was the case that a particular recipe called for a sequence of parts that could be obtained at this tower cluster, then a platform agent may be able to satisfy several intra-path precedence constraints in rapid succession with minimal time devoted to travel. Thus it may be beneficial for platform agents to reason not just about how to satisfy the next constraint during route search but also about the next several operations that need to be performed. Clustering algorithms could consider the sequence of operations required in a particular recipe and how to coordinate the actions of tower agents in a particular area to accomplish a sequence of constraints rather than single constraints. Such reasoning will likely result in increased computational load, but could potentially enhance solution quality substantially.

7.4.3 Tiered auctions with instantaneous task allocation

The tiered auction approach with instantaneous task allocation will factor in the results of route search during task allocation. In this approach the methods used to bound individual route searches and also to compare among different potential order assignments for the agents become crucial for maintaining tractability. An approach should still be able to exploit spatial structure in limiting the space of possible routes and bids that need to be explored. For instance, for a given order with a set of recipes, an agent should be able to determine the shortest possible path that can result in the successful construction of the task given that all parts can be provided with no delay. This should provide an upper bound on the quality that can be obtained from using a particular recipe for a particular task.

Furthermore, constraint clustering can potentially be exploited in jointly considering the space of potential order assignments, recipes, and routes. Many different orders and recipes may share sequences of components for assembly. For instance, if desktop computers were the objects being manufactured, some of the components between two different orders may be different - motherboards or graphics cards - but some may be the same, like power supplies. If the agent can determine tower clusters that can supply sequences of components, those routes can potentially be applied across different orders.

7.4.4 Tiered auctions with time-extended task allocation

In the disaster response domain there was a substantial performance increase associated with extending the time horizon beyond single task allocations. The primary intuition we leveraged in designing heuristics is that in many cases reaching one group of tasks would allow an agent to address other tasks in locations on or near the agent's route or near the destination fire. By using clustering and opportunistic path planning we exploited this spatial structure to tractably achieve longer time horizon coordination solutions. In terms of developing a similar technique in the on-order manufacturing domain, there are a few important differences which complicate the transference. First, while an agent's position in the environment can affect its fitness for a task, unlike in the disaster response domain tasks have no associated fixed location where they must be addressed. Objects can be constructed in many different locations using a variety of different methods. Thus disaster response heuristics that exploit the fact that tasks that are near each other can be efficiently addressed in succession do not have a clear analogy.

If platform agents can only carry one item for construction at a time, then time-extended allocation may have limited utility. Small performance gains may result when platform agents determine delivery locations on the basis of the next object they will construct, or when tower agents have more advanced notice for fetching desired items, but it is unlikely that large gains will result from reasoning about future allocations.

If platform agents can simultaneously manufacture several different objects, however, then time-extended allocation becomes essential in this case, as addressing a single order at a time will waste production capacity. The potential for constraint clustering becomes even greater when platform agents can potentially simultaneously be involved in constructing multiple items. In such domains reasoning about constraint clustering during multi-task allocation can potentially greatly improve performance. Agents can potentially be addressing tasks in parallel, and by moving to a given staging area can acquire the components to satisfy constraints for more than one object. In addition to reasoning about how a series of constraints can be satisfied by a tower cluster, multi-task allocation and planning should incorporate the observation that objects with similar recipes make good choices for multi-task allocations, as when there is commonality among necessary components and sequences for assembly then a single stop at a tower may allow recipes to be progressed on several different objects. Furthermore, depending on capacity constraints a tower agent may be able to fetch several of the same parts at the same time. Both clustering orders for assignment based on similar recipe structure and clustering constraint locations should enable tractable time-extended allocation.

7.4.5 Genetic algorithms for full coordination search

The genetic algorithm approach for full coordination search should be capable of fully searching the coordination space in the on-order manufacturing domain. Genomes will need to include not only the time-extended order assignment but also recipe selection, routes, and the order in which tower agents fetch components. Unlike the disaster response domain, where we represent routes instead of using an explicit task allocation, the GA representation for the on-order manufacturing domain will almost certainly need to explicitly represent allocations and recipes in addition to the routes taken to construct assigned orders. The actions that are performed at each stage in a route could potentially be used to assemble different items and different recipes, making a purely route based representation insufficient for encoding the full coordination space. Simulation for evaluation will need to account for the fact that the inventory of items available at staging areas is a product both of what tower agents fetch as well as what parts are acquired by platform agents and at what times.

Searching the full coordination space using a genetic algorithm required a good deal of time in the disaster response domain, but the time requirements will likely be even more substantial in the on-order manufacturing domain as not only must routes be represented, but also assignments and recipes, and tower agents have a potentially much larger possible space of sequences of items they could fetch. The size of the space means that heuristic operators will almost certainly need to be used to focus on the search on solutions that are likely to be of high quality. Potential heuristics could involve having tower agents tend to fetch the items that are required by the platform agents in the order that they are required by the selected platform recipes or using operators that reason about clustering.

7.4.6 Coping with uncertainty

The level of uncertainty in a factory setting will almost certainly be lower than in disaster response domains, and designers can exert a good deal of control over the environment. The most likely form of uncertainty will be dynamic task issue. Whether or not reactive re-planning will be a benefit in such domains will depend on whether or not platform agents can interrupt object manufacture once initiated. If plans can be interrupted, and there is a possibility that new orders may be substantially more important or urgent than current orders, then a re-planning strategy that assesses whether or not foregoing current assignments would be beneficial should improve the responsiveness of the system.

7.4.7 Accommodating domain variations

We already briefly touched on one variation of the on-order manufacturing domain, a variation where platform agents can simultaneously construct multiple items. Another potential variation of the domain involves another form of intra-path constraint. In this variation platform agents are not capable of attaching parts to objects; instead, tower agents fulfill a dual role of both fetching parts and attaching them to the objects. In this variation in addition to the precedence intra-path constraint that an object must be fetched before it can be attached to an object there is an additional simultaneity intra-path constraints, as the tower agents must co-locate with the platform agents in order to attach parts. As described in Section 7.2, this form of constraint serves to more strongly inter-constrain agent schedules but does not necessarily make determining a good coordination solution more difficult, as introducing simultaneity constraints reduces the space of potential solutions.

A variation that could potentially make the problem substantially more complex are domains that involve simultaneity requirements between platform agents, or where rendezvous can potentially be beneficial. This sort of constraint could exist if platform agents were not generally capable in terms of performing the operations associated with a recipe. Instead, to manufacture objects would require one platform agent to perform a series of operations to an object, and then to rendezvous with another platform agent with a different set of capabilities, which could continue the manufacture process.

7.5 Summary and discussion

In this chapter we extend our work to domains other than disaster response with intra-path precedence constraints. In the first part of the chapter we consider adding different kinds of simultaneity constraints to the disaster response domain. The first kind of intra-path simultaneity constraint we consider are simultaneity requirements between constraint-addressing agents at constraint sites. We showed that we could easily extend our market-based techniques to accommodate such constraints by centralizing arrival times so that auctioneers can determine completion times given that the efforts of multiple agents are required. We argued that the earliest completion time heuristic for constraint allocation and sequencing remains an effective method in allocating constraints with simultaneity requirements. The GA required more substantial adaptation, as it must search in the space of arbitrary assignments and sequences, which is made much more difficult by the incorporation of simultaneity constraints. In order to facilitate schedule synchronization between constraint addressing agents we introduced a new heuristic operator, SimulSwap. In experiments we showed that the market-based techniques still offer a good balance of performance and minimized computation. The GA approach has more difficulty searching in the full coordination space given the more dense interdependencies between agents schedules and fails to achieve the same performance as the TA-TE approach even with new heuristics and substantially more generations for optimization.

Next we argued that our approaches could support simultaneity between task-addressing and constraint-addressing agents at constraint sites, as such constraints drastically limit the space of potential reasonable constraint sequences. We then moved on to addressing domains with constraints between task-addressing agents at task sites. A general treatment of the problem requires considering the full multi-agent path planning problem; however, we observed that task-addressing agent coalitions, once co-located, could be treated as single agents in terms of coordination as long as maintaining a fixed coalition structure was appropriate for a domain. While achieving co-location is a potentially difficult problem, if co-location needs to happen rarely and agents are near each other then simple heuristic techniques are sufficient to achieve reasonable performance. Once agents are co-located and can be treated as a single unit, any of our market-based approaches or the GA approach could be used to determine coordination solutions. This treatment, however, assumes that the question of what coalitions to form is trivial - if sub-teams must form and separate regularly, then rendezvous and potentially considering different coalition structures become increasingly important components of the problem, requiring techniques that go beyond our suite of approaches. The problem of determining time-extended coalition formations in domains with intra-path constraints is a problem we will consider as future work.

In the second part of the chapter we extended our purview in a different direction - towards the potential applications of our suite of techniques to collaborative resource acquisition domains. While the specific features in such domains are considerably different than in domains where constraints must be addressed to allow passage, we argue that the central intuitions associated with our suite of approaches would extend to resource acquisition domains.

Chapter 8 Conclusion

Determining coordination solutions in domains with intra-path constraints is an important research problem that has not been adequately addressed by the multi-agent research coordination community. Intra-path constraints are inter-agent constraints that must be satisfied to allow agents to reach particular locations in the environment; planning for constraint satisfaction in such domains is required in order to determine sets of agents plans that will minimize delay and maximize solution quality. Reasoning about the effect that agent schedules will have on other agent's path plans makes domains with intra-path constraints especially difficult. Existing methods for determining coordination solutions are not equipped to tractably determine high quality coordinated behavior given the potentially large search spaces for coordination.

This dissertation explores a number of different approaches for determining coordination solutions in domains with intra-path constraints. Our goal is not to suggest an approach that is appropriate in all scenarios, but to provide a range of different approaches with different strengths and weaknesses and to articulate how different approaches will perform given different scenarios. In developing each of the different approaches our goal is to exploit domain characteristics to develop heuristics and bounding methods so as to improve solution quality while limiting the computational requirements for search.

We first summarize our findings in terms of approach strengths and weaknesses in terms of scenario factors. We then state the contributions this thesis makes to the literature of multi-robot coordination. Finally, we present future work suggested by this dissertation.

8.1 Approaches in the context of scenario factors

In this section we summarize our findings of approach performance in terms of a variety of the scenario factors we introduced in Chapter 1.3. The goal is to encapsulate our results so that approach

Approach	High constraint density	Limited constraint-addressing resources	High task durations	Simultaneity constraints at constraint sites
Ind. Tasks	– –	–	+	
ATC	–		–	
TA-IA		+	+	
TA-TE		–	–	
GA			–	–

Table 8.1: General guidelines on the strengths and weaknesses of the approaches in static domain instances with different characteristics. Strengths are denoted with the + symbol, and weaknesses with the – symbol, with particular strengths or weaknesses doubling these labels.

designers can easily determine the strengths and weaknesses of different approaches in different environments. We first consider solution quality as a product of factors within the context of static domain instances. Next we consider approach performance with different kinds of uncertainty. Finally, we consider performance in terms of scenario resources and requirements.

8.1.1 Static domain factors

In this work the primary way we capture relative approach performance is through the gaps between attained reward in terms of the objective function. With a few exceptions, our performance hierarchy remained largely intact throughout experimental testing. However, we showed that the gaps extended or narrowed given different parametrizations that affect the nature of domain instances. Of particular interest for each approach are the performance gaps between approaches that are adjacent on the performance hierarchy. It is unlikely, for instance, that an approach designer would be choosing between deploying the independent tasks approach or the genetic algorithm approach, as the differences between the two approaches in terms of scenario requirements are vast. A designer is much more likely to be choosing between independent tasks and ATC, or considering whether moving from TA-IA to TA-TE will improve solution quality enough to justify the potentially greater cost in terms of computation. We try to capture whether a designer should be more or less likely to choose a particular approach if the domain in question has a certain characteristic or the scenario a particular requirement. In Table 8.1 we show results for static domain factors. Strengths are denoted with the + symbol, and weaknesses with the – symbol, with particular strengths or weaknesses doubling these labels.

We showed in Chapter 5.2.1 that our minimum time approaches perform relatively poorly in domains with high densities of intra-path constraints, as the allocation strategies do not consider how inter-agent constraints affect allocation quality. In Chapter 5.2.2 we showed that two approaches fared relatively poorly in domains with limited constraint-addressing resources: the independent tasks approach performs poorly for reasons similar to those that cause poor performance in domains

Approach	Dynamic task issue	Short sensing ranges	RoboCup Rescue levels of uncertainty
Ind. Tasks	–	+	+
ATC	–		
TA-IA	+	++	–
TA-TE		– –	– –
GA	–	– –	– –

Table 8.2: General guidelines of the strengths and weaknesses of the approaches in dynamic instances with different characteristics.Strengths are denoted with the + symbol, and weaknesses with the – symbol, with particular strengths or weaknesses doubling these labels.

with intra-path constraints. The relatively poor performance of the TA-TE approach can be attributed to allowing the over-exploitation of constraint-addressing resources. For domains with high task duration we showed in Chapter 5.2.3 that performance gaps narrowed considerably across all approaches; however, the independent tasks approach and the TA-IA approach achieved relatively high performance in comparison to the other approaches. Finally, in Chapter 7.3.2 we considered adding simultaneity constraints between constraint-addressing agents. While the market-based approaches handled this readily, even with the development of new heuristics the GA approach performed relatively poorly.

8.1.2 Coping with uncertainty

Table 8.2 summarizes our findings in domains with dynamic task issue. These results reflect the re-planning adaptations we made to the approaches. We presented results for dynamic task issue in Chapter 6.1.4. Our primary findings in this domain were that the minimum time approaches performed relatively poorly, largely due to the fact that delay incurred at constraint sites is more costly in domains with dynamic task issue. The TA-IA approach does a good job of balancing reactivity and planning horizon even without incorporating reactive re-planning, and the performance gap between the TA-IA and TA-TE approaches narrow even when the TA-TE approach is equipped with a partial re-planning strategy. We equipped the GA approach to cope with uncertainty, but doing so requires even more computational time. In domains with uncertain intra-path constraint locations and short sensing ranges, we showed in Chapter 6.2.1 that the more reactive TA-IA approach, especially when supplemented with a reactive re-planning achieved much higher performance than the TA-TE approach. The less reactive, longer planning horizon approaches will perform poorly in domains with this level of uncertainty. Finally we recognized that in domains with extremely high levels of uncertainty, such as the RoboCup Rescue Simulation League, implicitly scheduled coordination, or some combination of explicitly and implicitly scheduled coordination may offer superior performance to any of our explicitly scheduled approaches. We leave this evaluation for

Approach	Little time for planning	Parallelizability	Distributability	Communication cost
Ind. Tasks	++	+	++	++
ATC	+			+
TA-IA				–
TA-TE	–			–
GA	––	++	––	

Table 8.3: General guidelines of the strengths and weaknesses of the approaches in terms of scenario resources and requirements. Strengths are denoted with the + symbol, and weaknesses with the – symbol, with particular strengths or weaknesses doubling these labels.

future work.

8.1.3 Resource and requirements

Table 8.3 summarizes our findings in terms of resource requirements. A primary focus of our work has been determining the cost in total computation time for determining coordination solutions. For scenarios with little time for planning, one of our two minimum time algorithms – independent tasks or ATC – are most appropriate. The TA-IA approach is generally at least an order of magnitude more expensive in computational terms. The TA-TE approach is of the same order of magnitude in terms of computational costs as the TA-IA approach, but in most domains will be somewhat more expensive. The parametrizations of the GA that perform as well as the TA-TE approach take at least two orders of magnitude more time to do so, and the required computation increases when re-optimization must occur due to uncertainty.

The next three scenario requirements were not quantitatively addressed in this work; doing so is a matter for future work. We can still draw some general conclusions, however, based on approach design. In terms of parallelizability we particularly care about the potential to reduce the wall-clock time required for computation by spreading that computation out among a number of different processors. This should be largely possible in the independent tasks approach, as the most costly computation is in agents determining bids, a process that could be conducted in parallel. For the approaches that involve route search, parts of the computation would be easy to parallelize and parts would be difficult. Computation among agents competing for auction tasks can be parallelized, both for task-addressing agents and constraint-addressing agents. However, route search for individual task-addressing agents may be difficult to parallelize. For the GA approach parallelization would be of great benefit, as the most expensive part of the computation, simulation to determine fitness, can be easily computed in parallel by different processors.

In terms of distributability substantial work remains in determining how well our approaches would extend to a distributed framework. Distributed versions of our market-based approaches would suppose that each agent represented a physically separate robot with its own perception of the world state. Aspects we presented as function calls, like bidding functions, would require communication over whatever network was in place. The independent tasks approach should be highly distributable, as agent bids can be determined in isolation. What effect using a distributed approach would have on our explicitly scheduled market-based approaches is an open question. If the effect was just of introducing minor durational uncertainty then the approaches could be effective. If the domain is a highly uncertain one where individual agents could have very different perceptions of the world then using a distributed approach would be likely to undermine the ability of agents to form accurate explicitly scheduled coordination solutions. We leave this question for future work.

Finally, we briefly touched on the cost of communication, an important factor for some scenarios. The independent tasks approach should require the least communication in a distributed version, with the ATC version the next least as it only conducts a single route search per assignment. The tiered auction approaches would be quite expensive in communication terms, as during a single auction cycle many different sub-auctions for different task-addressing agents, tasks, and routes may be conducted. The genetic algorithm requires information centralization.

8.2 Contributions

In this section we summarize the contributions this work has made to the field of multi-robot coordination.

Contribution 1: A suite of approaches for determining effective coordination solutions in domains with intra-path constraints The primary contribution of this thesis is a suite of approaches to coordinating agents in domains with intra-path constraints. We recognize that different scenarios may have a variety of factors that go into approach design: characteristics of the domain, the level of uncertainty, resources available for determining solutions, and requirements for evaluating approach performance. Our approaches have different strengths and weaknesses in terms of required computation time, the quality of solutions determined in domains with different characteristics, and reactivity to new information. In each approach we exploit domain features to develop heuristics and bounding approaches that serve to improve the quality of the determined coordinated behavior while limiting computational requirements. Our suite of approaches will perform well in scenarios with a variety of different characteristics and requirements.

Contribution 2: The first formulation of domains with intra-path constraints as a distinct problem space within multi-agent coordination and the first comprehensive treatment of the factors that must be addressed in approach design While some previous work has been applicable to domains with intra-path constraints, no work has recognized that a number of applications can be grouped in this categorization or formulated approaches that are applicable in a number of different domains. In this dissertation we described how the presence of intra-path constraints creates a coordination space where some agent routes must be included, adding levels of complexity to coordination space hierarchies. Furthermore, we articulate the different factors that can potentially affect approach performance and thoroughly evaluate our approaches in regard to these factors, offering guidelines for approach designers.

Contribution 3: The first use of tiered auctions to allow agents to consider and allocate agent interactions during the auction process Tiered auctions are a novel technique we developed initially for instantaneous allocation in a tightly coordinated domain (Jones *et al.*, 2006). In this dissertation we have used them extensively as a method that allows agents to solicit the involvement of other agents in forming a plan to address tasks. This method allows agents to form accurate, explicitly scheduled coordination solutions involving both task-addressing and constraint-addressing agents which can be factored into allocation decisions to improve solution quality.

Contribution 4: The first tractable, time-extended allocation approach in domains with intra-path constraints The size of the coordination spaces in domains with intra-path constraints are so large that even single task allocation can be difficult, and in order to cope with the exponentially larger space of multi-task allocations is a substantial challenge. By making extensive use of domain-specific features we have developed bounding and heuristic approaches that allow us to determine high quality time-extended coordination while incurring only a small increase in computational cost over an instantaneous allocation approach.

Contribution 5: The first application of genetic algorithms to domains with intra-path constraints We are aware of no existing work that applies genetic algorithms to domains where allocation and path-planning are inter-constrained, as they are in domains with intra-path constraints. Our approach represents a novel application of genetic algorithms.

8.3 Future work

There are a number of different avenues for future research building from the work in this dissertation. We divide potential future work into several different categories.

8.3.1 Extending our implementation and evaluation methods

The existing implementation of our approaches is designed to make all coordination decisions within a single process on a single machine. As described earlier in this chapter, one avenue for future work is to extend our implementation to support parallelism or distribution. A parallel version of certain elements of the approaches should be straight-forward to implement; using such an approach should allow speed-ups in wall clock terms by dividing the computational load among a number of different cores or processors. Obvious candidates for parallelization would be in the route search processes for agents in the auction based approaches and in simulation during genetic algorithm evaluation. For the auction-based approaches it should be possible to implement a distributed version, where each agent is assumed to have a distinct, physically separated world view and processor. Developing a distributed approach is a substantial effort, as it requires confronting the issues posed by potentially asynchronous and unreliable communication and uncertainty in local the perceptions of world state.

Extending the implementation should permit evaluation by a variety of new metrics. First, for parallel implementations wall-clock time becomes a potentially more important metric than total computation. For distributed implementations, especially those that involve robotic hardware, execution time becomes a factor. In domains where planning and execution need to occur in tandem, an interesting metric for evaluation becomes how much planning can be interleaved with execution. The overall amount of computation, or even wall-clock computation time may become less important than determining when the computation needs to occur. The goal may become to minimize the up-front planning cost so that execution can begin, but there may be additional time to continue to refine the solution using computational resources that are not required for execution.

8.3.2 Adaptive constraint satisfaction pricing

One potential difficulty in the implementation of our greedy, market-based approaches is in preventing task-addressing agents from monopolizing constraint-addressing resources. Agents in the market-based system search the coordination space based on maximizing individual reward, and are given free access to assigning constraint-addressing agents to constraints. The result can be that in order to obtain potentially marginal increases in reward agents may monopolize constraint-addressing resources, when it could potentially be better in terms of maximizing overall solution quality to spread constraint-addressing resources among task addressing agents. This is especially a problem when using tiered-auctions with time-extended allocation, as task-addressing agents may plan far into the future. We can limit this monopolization by setting the NUM_DEBRIS_CONS to limit the permissible number of constraints that can lie on a path and tightening the g-value bounding to limit path indirection, but determining these parameter values for domains with different characteristics can be difficult.

A better approach than depending solely on parameter tuning may be a strategy that implicitly represents some notion that monopolizing resources can be costly. The goal of such an approach would be to equip agents to determine what magnitude of reward increase justifies using additional resources. One potential method for enabling this reason is to incorporate some notion ofopportunity cost for using constraint-addressing resources, and to reason about these costs during the bidding and route search processes. For instance, constraint-addressing agents could charge a certain amount for each unit of time they need to spend to address a particular constraint. Setting such a cost would allow task-addressing agents to reason about the costs of using extra resources versus the marginal benefits in terms of rewards. The main research in implementing such an approach is how to determine costs in order to improve solutions. This is a difficult problem, as costs should almost certainly scale with demand. In some cases, particularly when constraint-addressing resources are plentiful, it may beneficial to have task-addressing agents be as greedy as possible, whereas when resources are scarce costs should be high to discourage wanton use. A system for adaptive costs for taking on constraint satisfaction duties can improve performance, but the exact methods for doing so remain an interesting resource challenge.

8.3.3 Confronting real world uncertainty

In this dissertation we took a high-level view in terms of handling uncertainty, examining how particular, easily modeled forms of uncertainty can be accommodated within our approaches and how they affect performance. If the agents in a domain are physically-embodied robots, uncertainty is likely to permeate all levels of the system, including perceptual uncertainty, communication uncertainty, and uncertainty in action. This uncertainty is likely to make forming accurate explicitly coordinated schedules difficult, requiring frequent re-planning to account for the changing world state. One potential avenue of research could be in merging some elements of implicit scheduling into the process of determining explicitly scheduled coordination. Such reasoning could incorporate statistical data in adjusting bid valuations, a research direction we considered in independently coordinated domains (Jones *et al.*, 2007). Furthermore, an active area of research in scheduling is how to anticipate uncertainty to make more flexible plans (Sellner & Simmons, 2008). We could potentially extend our methods to reason about contingencies and coping with durational uncertainty using such methods.

8.3.4 Full coordination search

While our genetic algorithm approach can outperform heuristic approaches if given sufficient computation time, there is more work to be done in speeding up the approach and ensuring that the approach can avoid local maxima and determine solutions that are near optimal. One avenue of future research is in developing better heuristics that could potentially speed up search ideally while still avoiding the local performance maxima that can result from using more heuristic approaches. Secondly, our approach does full coordination search of both task-addressing and constraint-addressing schedules simultaneously. Another approach would be to structure the search more hierarchically. For instance, we could potentially do a separate optimization of constraint-addressing schedules each time a change is made to task-addressing agent schedules. Whether this approach would improve performance is a matter for further study. Another potential improvement would be to develop a more implicit, co-evolution strategy, such as the one used by Cai and Peng

(Cai & Peng, 2002). The co-evolution strategy would attempt to separately evolve populations of high-performing task- and constraint-addressing agent schedules in the hopes that the results could be merged into a high-performing full coordination solution. While this approach would be unlikely to replace full optimization it could provide a better method of genome initialization.

8.3.5 Time-extended coalition formation

As mentioned in Chapter 7, the problem of how to tractably determine time-extended solutions involving adaptive coalition structure is a problem we do not fully address in this work. Determining what coalitions will offer superior performance in a time-extended fashion requires reasoning about costs or delays of forming and breaking coalitions as well as potential benefits. Within an explicit scheduling framework determining costs and benefits becomes especially difficult, as an approach must consider not only the space of solutions associated with the current coalition structure, but the potential space of solutions associated with altering the coalition structure. One potential solution approach would be to incorporate an implicit mechanism that would attempt to identify beneficial coalition structures without explicitly scheduling changes in coalition structure.

Bibliography

Babanov, A., and Gini, M. 2006. Deciding task schedules for temporal planning via auctions. In *AAAI Spring Symposium: Distributed Plan and Schedule Management.*

Bererton, C. 2004. *Multi-robot coordination and competition using mixed integer and linear programs.* Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Berhault, M.; Huang, H.; Keskinocak, P.; Koenig, S.; Elmaghraby, W.; Griffin, P.; and Kleywegt, A. 2003. Robot exploration with combinatorial auctions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Botelho, S. S. C., and Alami, R. 1999. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

Bowling, M.; Browning, B.; and Veloso, M. 2004. Plays as effective multiagent plans enabling opponent-adaptive play selection. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS'04).*

Cai, Z., and Peng, Z. 2002. Cooperative coevolutionary adaptive genetic algorithm in path planning of cooperative multi-mobile robot systems. *Journal of Intelligent and Robotic Systems* 33:61–71.

Collins, J.; Sundareswara, R.; Tsvetovat, M.; Gini, M.; and Mobasher, B. 1999. Search strategies for bid selection in multi-agent contracting. In *IJCAI Workshop on Agent-Mediated Electronic Commerce.*

Collins, J.; Gini, M.; and Mobasher, B. 2002. Multi-agent negotiation using combinatorial auctions with precedence constraints. Technical Report 02-009, University of Minnesota, Department of Computer Science and Engineering.

Cordeau, J.-F., and Laporte, G. 2007. The dial-a-ride problem: models and algorithms. *Annals of Operations Research 153* 153:29–46.

Dias, M. B.; Zinck, M.; Zlot, R.; and Stentz, A. 2004a. Robust multirobot coordination in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

Dias, M. B.; Zlot, R.; Zinck, M.; Gonzalez, J. P.; and Stentz, A. 2004b. A versatile implementation of the *traderbots* approach to multirobot coordination. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS).*

Dias, M. B. 2004. *TraderBots: a new paradigm for robust and efficient multirobot coordination in dynamic environments.* Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University.

Enright, J. J.; Savla, K.; Frazzoli, E.; and Bullo, F. 2009. Stochastic and dynamic routing problems for multiple UAVs. *AIAA Journal of Guidance, Control, and Dynamics* 34(4):1152–1166.

Erol, K.; Hendler, J.; and Nau, D. 1994. Semantics for hierarchical task network planning. Technical Report CS-TR-3239, UMIACS-TR-94-31, Computer Science Dept., University of Maryland.

Gerkey, B. P., and Matarić, M. J. 2002. Sold!: auction methods for multi-robot control. *IEEE Transactions on Robotics and Automation Special Issue on Multi-Robot Systems* 18(5).

Gerkey, B. P., and Matarić, M. J. 2004. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research* 23(9).

Gerkey, B.; Thrun, S.; and Gordan, G. 2004. Visibility-based pursuit-evasion with limited field of view. In *Proceedings of the National Conference on Artificial Intelligence (AAAI).*

Goldberg, D. E. 1989. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley Professional.

Gross, D., and Harris, C. M. 1998. *Fundamentals of queueing theory.* New York: Wiley, 3rd edition.

Guerrero, J., and Oliver, G. 2006. Physical interference impact in multi-robot task allocation auction methods. In *IEEE Workshop on Distributed Intelligent Systems.*

Hollinger, G., and Singh, S. 2008. Proofs and experiments in scalable, near-optimal search by multiple robots. In *Robotics: Science and Systems (RSS).*

Hollinger, G.; Kehagias, A.; and Singh, S. 2007. Probabilistic strategies for pursuit in cluttered environments with multiple robots. In *IEEE International Conference on Robotics and Automation.*

Hunsberger, L., and Grosz, B. J. 2000. A combinatorial auction for collaborative planning. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS).*

Jones, E. G.; Browning, B.; Dias, M. B.; Argall, B.; Veloso, M.; and Stentz, A. 2006. Dynamically formed heterogeneous robot teams performing tightly-coupled tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

Jones, E. G.; Dias, M.; and Stentz, A. 2007. Learning-enhanced market-based task allocation for oiver-subscribed domains. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Jones, E. G.; Dias, M. B.; and Stentz, A. 2008. Tiered auctions for multi-agent coordination in domains with precedence constraints. In *Proceedings of the 26th Army Science Conference.*

Jones, E. G.; Dias, M. B.; and Stentz, A. 2009. Time-extended multi-robot coordination in domains with intra-path constraints. In *Robotics: Science and Systems (RSS)*.

Jørgensen, R.; Larsen, J.; and Bergvinsdottir, K. 2007. Solving the dial-a-ride problem using genetic algorithms. *The Journal of the Operational Research Society*.

Kalra, N. 2006. *A market-based framework for tightly-coupled planned coordination in multirobot teams.* Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University.

Kitano, H.; Tadokoro, S.; Noda, I.; Matsubara, H.; Takahashi, T.; Shinjou, A.; and Shimada, S. 1999. Robocup Rescue: search and rescue in large-scale disasters as a domain for autonomous agents research. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 6.

Koes, M.; Nourbakhsh, I.; and Sycara, K. 2005. Heterogeneous multirobot coordination with spatial and temporal constraints. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Kwak, J.; Varakantham, P.; Taylor, M.; Marecki, J.; Scerri, P.; and Tambe, M. 2009. Exploiting coordination locales in distributed pomdps via social model shaping. *The AAMAS workshop on Multi-agent Sequential Decision-Making in Uncertain Domains.*

Lagoudakis, M. G.; Berhault, M.; Koenig, S.; Keskinocak, P.; and Kleywegt, A. J. 2004. Simple auctions with performance guarantees for multi-robot task allocation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Lagoudakis, M.; Markakis, E.; Kempe, D.; Keskinocak, P.; Kleywegt, A.; Koenig, S.; Tovey, C.; Meyerson, A.; and Jain, S. 2005. Auction-based multi-robot routing. In *Robotics: Science and Systems (RSS).*

Lemaire, T.; Alami, R.; and Lacroix, S. 2004. A distributed tasks allocation scheme in multi-UAV context. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

Lin, L., and Zheng, Z. 2005. Combinatorial bids based multi-robot task allocation method. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

MacKenzie, D. C. 2003. Collaborative tasking of tightly constrained multi-robot missions. In *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings of the 2003 International Workshop on Multi-Robot Systems*, volume 2. Kluwer Academic Publishers.

Melachrinoudis, E.; Ilhan, A. B.; and Min, H. 2007. A dial-a-ride problem for client transportation in a health-care organization. *Computers and Operations Research.*

Melvin, J.; Keskinocak, P.; Koenig, S.; Tovey, C.; and Ozkaya, B. Y. 2007. Multi-robot routing with rewards and disjoint time windows. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Mesghouni, K.; Hammadi, S.; and Borne, P. 2004. Evolutionary algorithms for job-shop scheduling. *International Journal of Applied Math and Computer Science* 14(1):91–103.

Mills-Tettey, A. 2008. Bounded optimal coordination for human-robot teams. Ph.D. Disseration Proposal.

Nair, R.; Ito, T.; Tambe, M.; and Marsella, S. 2002. Task allocation in the robocup rescue simulation domain: a short note. In *RoboCup 2001: Robot Soccer World Cup V*, 751–754. London, UK: Springer-Verlag.

Okamoto, S.; Scerri, P.; and Sycara, K. P. 2008. The impact of vertical specialization on hierarchical multi-agent systems. In *AAAI*, 138–143.

Paquet, S. 2006. *Distributed decision-making and task coordination in dynamic, uncertain, and real-time multiagent environments.* Ph.D. Dissertation, Université Laval.

Ropke, S.; Cordeau, J.-F.; and Laporte, G. 2007. Models and a branch-and-cut algorithm for pickup and delivery problems with time windows. *Networks* 49:258–272.

Russell, M. A., and Lamont, G. B. 2005. A genetic algorithm for unmanned aerial vehicle routing. In *Proceedings of the Genetic Engineering and Evolutionary Computation Conference (GECCO).*

Ryan, M. 2007. Graph decomposition for efficient multi-robot path planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).*

Sariel, S., and Balch, T. 2006. Dynamic and distributed allocation of resource constrained project tasks to robots. In *Multi-Agent Robotic Systems (MARS) Workshop at the Third International Conference on Informatics in Control, Automation and Robotics.*

Sariel, S.; Balch, T.; and Erdogan, N. 2007. Incremental multi-robot task selection for resource constrained and interrelated tasks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Scerri, P.; Farinelli, A.; Okamoto, S.; and Tambe, M. 2005. Allocating tasks in extreme teams. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 727–734. New York, NY, USA: ACM Press.

Schurr, N.; Marecki, J.; Tambe, M.; Scerri, P.; Kasinadhuni, N.; and Lewis, J. 2005. The future of disaster response: Humans working with multiagent teams (without being overwhelmed). In *Proceedings of AAAI Spring Symposium on AI Technologies for Homeland Security.*

Sellner, B., and Simmons, R. 2008. Duration prediction for proactive replanning. In *Proceedings of the IEEE International Conference on Robotics and Automation.*

Shima, T.; Rasmussen, S.; Sparks, A.; and Passino, K. 2006. Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms. *Computers and Operations Research* 33:3252–3269.

Simmons, R.; Apfelbaum, D.; Burgard, W.; Fox, D.; Moors, M.; Thrun, S.; and Younes, H. 2000a. Coordination for multi-robot exploration and mapping. In *Proceedings of the National Conference on Artificial Intelligence (AAAI).*

Simmons, R.; Singh, S.; Hershberger, D.; Ramos, J.; and Smith, T. 2000b. First results in the coordination of heterogeneous robots for large-scale assembly. In *Proceedings of the International Symposium on Experimental Robotics (ISER).*

Sims, M.; Mostafa, H.; Horling, B.; Zhang, H.; Lesser, V.; and Corkill, D. 2006. Lateral and hierarchical partial centralization for distributed coordination and scheduling of complex hierarchical task networks. In *AAAI 2006 Spring Symposium: Distributed Plan and Schedule Management.*

Smith, S. L., and Bullo, F. 2009. The dynamic team forming problem: throughput and delay for unbiased policies. *Systems and Control Letters.* (Submitted Jul 2008) to appear.

Smith, S.; Gallagher, A. T.; Zimmerman, T. L.; Barbulescu, L.; and Rubinstein, Z. 2007. Distributed management of flexible times schedules. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems.*

Smith, S. L.; Pavone, M.; Bullo, F.; and Frazzoli, E. 2009. Dynamic vehicle routing with priority classes of stochastic demands. *SIAM Journal on Control and Optimization.* Submitted.

Stroupe, A. 2003. *Collaborative execution of exploration and tracking using move value estimation for robot teams (MVERT).* Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Suàrez, S.; Collins, J.; and Lòpez, B. 2005. Improving rescue operations in disasters: approaches about task allocation and re-scheduling. In *The 24rd Annual Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG)*, 66–75.

Sultanik, E.; Modi, P. J.; and Regli, W. 2007. On modeling multiagent task scheduling as a distributed constraint optimization problem. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).*

Tang, F., and Parker, L. 2007. A complete methodology for generating multi-robot task solutions using AsyMTRe-D and market-based task allocation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

Tang, F., and Saha, S. 2008. An anytime winner determination algorithm for time-extended multi-robot task allocation. In *International Conference on Automation, Robotics and Control Systems*, 123–130.

Toth, P., and Vigo, D., eds. 2001. *The vehicle routing problem.* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Vail, D., and Veloso, M. 2003. Multi-robot dynamic role assignment and coordination through shared potential fields. In Schultz, A.; Parker, L.; and Schneider, F., eds., *Multi-robot systems: from swarms to intelligent automata: proceedings from the 2003 international workshop on multi-robot systems*, volume 2. Kluwer Academic Publishers.

Vig, L., and Adams, J. A. 2006. Multi-robot coalition formation. In *IEEE Transactions on Robotics*, volume 22 (4).

Zhang, X.; Lesser, V.; and Abdallah, S. 2005. Efficient management of multi-Linked negotiation based on a formalized model. *Autonomous Agents and Multi-Agent Systems* 10(2):165–205.

Zhang, V. L. X. 2002. Multi-linked negotiation in multi-Agent system. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 1207–1214.

Zheng, X., and Koenig, S. 2008. Reaction functions for task allocation to cooperative agents. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems.*

Zimmerman, T. L.; Smith, S.; Gallagher, A. T.; Barbulescu, L.; and Rubinstein, Z. 2007. Distributed management of flexible times schedules. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems.*

Zlot, R.; Stentz, A.; Dias, M. B.; and Thayer, S. 2002. Multi-robot exploration controlled by a market economy. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

Zlot, R. 2006. *An auction-based approach to complex task allocation for multirobot teams.* Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University.