On-Line Planning of Flexible Assembly Systems: An Agent-Based Approach

Cheng-Hua Wang Christiaan J.J. Paredis

Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890
{cwwang, cjp}@ices.cmu.edu

Abstract

This paper presents an agent-based approach to online assembly task allocation and scheduling for flexible assembly systems. We propose an anytime-scheduling algorithm to handle dynamic changes during planning. This algorithm is the center of an agent-based architecture in which agents for scheduling, collision avoidance, task execution, and task monitoring collaborate to overcome unexpected execution errors. A prototype planning system has been implemented for the RobotWorld flexible assembly system. Preliminary simulation results show that our approach is capable of dynamically rescheduling assembly operations after system faults occur. We are currently integrating the planning system with the assembly test bed in our laboratory.

Keywords: on-line planning, assembly planning, anytime algorithm, agent-based systems, flexible assembly systems.

1 Introduction and Related Work

To stay competitive in the current marketplace, companies must be able to react quickly to changing demands and environments. Traditional fixed automation systems can only handle specific types of products with high demand volumes. To manufacture small-batch and high-variety products effectively, we need flexible automation systems that can be easily reconfigured and programmed [9].

Because of its computational complexity, process planning is usually performed off-line. However, on-line programming is important for flexible assembly systems that need to be able to adapt quickly to new products or product mixes, and overcome unexpected system faults to avoid costly down-time.

To address the on-line planning issue, Boddy and Dean [3] propose the term *"Anytime Algorithm"* in their work on time-dependent planning to indicate algorithms that can return the best solution so far at any given time. The nature of on-line planning approaches is appropriate and practical for flexible manufacturing and assembly environments, because it enables load balancing between planning and execution. Anytime algorithms have been used in other complex and time-critical planning and decision-making problems such as robot control (sensor interpretation and path planning) [19] and database query [16].

Agent-based approaches have been widely used in manufacturing because they have the advantage to be modular, scalable, reconfigurable, distributed, easy to implement and maintain, and fault-tolerant. In particular, several researchers have addressed the use of multiple agent system for the control of robotic assembly systems. Basran [2] proposes a contract-net protocol for negotiation and dynamic task allocation in flexible assembly cells. Oliveira [12] presents a cooperative multi-agent system for robotic assembly cells using a blackboard architecture as an inter-agent communication mechanism. Rizzi et al [14] propose an architecture for a highly flexible, modular, and distributed precision assembly system. Other approaches in which agents are associated either with robots or with the parts the robots manipulate have been proposed in [10], [11] and [13].

Assembly planning can be divided into task decomposition, task allocation, and task execution. In the task decomposition stage, a preliminary plan is generated that consists of a sequence of assembly operations and precedence relationships among them. Various researchers have studied assembly sequence generation [7][17] based on accessibility, stability, and design for assembly (DFA) rules. At this stage, only high-level task commands are considered. During the task allocation stage, this high level assembly plan is then converted into low-level robot commands that can be executed by the assembly hardware. The necessary resources are assigned to every assembly operation after which they are scheduled for execution, during the task execution stage.

In practice, there is always a discrepancy between planning and execution results. At the planning stage, one has to rely on models describing the system behavior. These models are only approximations of the physical reality. Especially in

multi-robot systems, it is often prohibitively expensive to model all the interdependencies between the system components. For instance, predicting the time required to move from point A to point B may already prove challenging in multi-robot systems in which one of the robots may have to slow down to avoid a collision with another robot. In our approach, sensors monitor the task execution and provide feedback, so that the planner can adjust its schedule dynamically.

2 Problem Definition

Traditional assembly planning systems produce assembly plans prior to execution, in an off-line fashion. They assume that the assembly environment is static and predictable. However, as illustrated above, these assumptions are inappropriate for flexible assembly systems. Therefore, we are developing an on-line planning system for flexible robotic assembly that addresses planning, scheduling, and control in real-time.

In our assembly planner we consider the task execution stage in addition to task decomposition and task allocation. However, the focus of this paper is on on-line assembly task allocation and scheduling.

In our previous work on *Intelligent Assembly Modeling and Simulation* (IAMS) [4], we have developed a software environment for modeling and simulating the complete assembly process: design and editing of assemblies, assembly-sequence generation, assembly tool selection, and assembly process simulation. The focus of IAMS was on assembly task decomposition, modeling, and simulation. To extend the capability of the IAMS framework, we have now developed an on-line assembly planning system for assembly task allocation and scheduling. The planning system is capable of allocating resources and generating operation schedules according to dynamic changes such as manipulator breakdowns. It also adjusts its planning strategies based on feedback from a task execution monitor.

Our current system uses an agent-based approach based on the architecture developed in our previous work [8]. Agent-based approaches have been used widely in flexible manufacturing and assembly environments. Due to their distributed nature, they are better suited than centralized approaches for dealing with dynamic and unexpected changes.

As is illustrated in Figure 1, the inputs to our planning system are the geometric models of the product and a precedence graph defining the set of possible assembly sequences. The output is a dynamic schedule for a flexible assembly cell, that is, low-level motion commands for each of the robot manipulators in the system. This schedule depends on the task execution and is dynamically updated when there are discrepancies between the system execution and the internal model.

3 Anytime Scheduling Algorithm

Solving a scheduling problem requires allocating resources and determining a task sequence [1]. Scheduling problems are often viewed as constrained optimization problems. One can optimize the schedule with respect to several possible performance criteria, such as flow time or total cost. The constraints include, for instance, precedence constraints and resource utilization constraints.

Static scheduling algorithms produce schedules ahead of time with the assumption that every activity is known in advance. Although these algorithms may be able to compute an “optimal” schedule, in

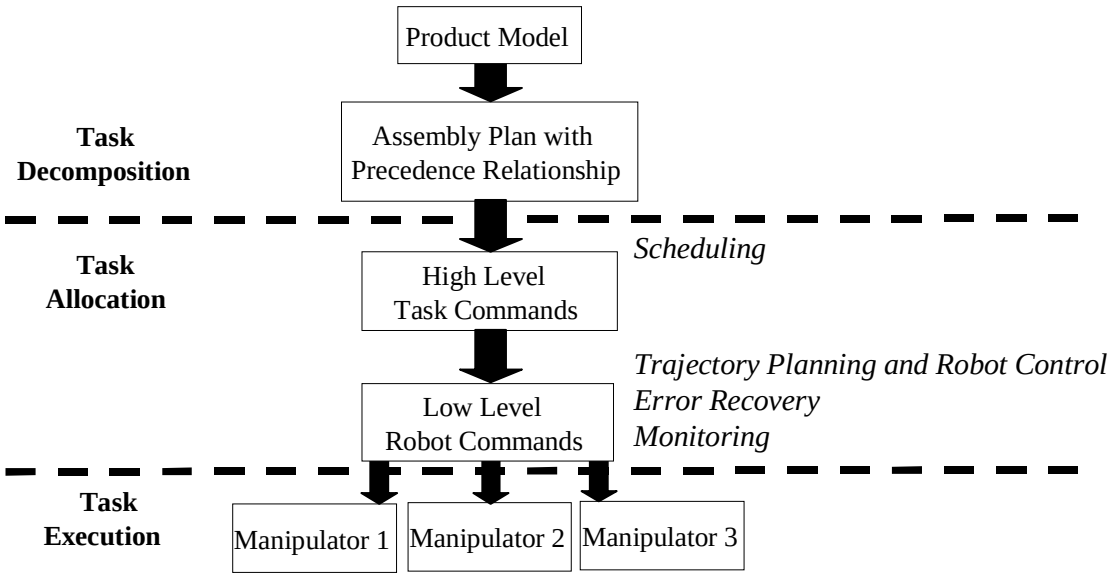


Figure 1: Assembly System Overview

practice they rarely achieve optimality due to the computational complexity of the scheduling problem. Another pitfall of static scheduling is that it does not work well in dynamically changing environments such as flexible assembly systems. Any time the environment changes, the schedule has to be recomputed from scratch.

On-line scheduling approaches (for instance, [18]), can solve dynamic scheduling problems much more gracefully. One example of such an on-line scheduling approach is based on *Anytime Algorithms*. Anytime scheduling algorithms are algorithms that can provide a feasible schedule at any time, while improving this solution over time.

We propose an anytime scheduling algorithm based on the A* search algorithm [6]. A* is a best first heuristic search algorithm that explores the nodes with minimal cost $f(n) = g(n) + h(n)$ first; n is the current node, $g(n)$ is the cost from the initial node to current node, and $h(n)$ is the estimated cost from current node to the goal node. It guarantees to return a minimum cost solution as long as the heuristic, $h(n)$, does not overestimate the remaining cost.

The objective of our scheduling algorithm is to minimize total operation time. The operation time is estimated based on the characteristics of the operation and the robot, e.g. the distance to be traveled, and the maximum velocity and acceleration. The algorithm also tries to schedule as many different resources as possible to achieve load balancing. It contains the following three steps:

1. The search starts without any scheduled assembly tasks.
2. Subject to the precedence constraints, unscheduled tasks are added to the search path, one at a time.
3. For each newly added task in the path, all available resources (robots) are queried for an estimated operation time. This estimate is used to compute the heuristic cost function, $h(n)$.
4. The search continues until an optimal solution (with minimum total operation time) is found; if no feasible solution exists, a failure is reported.

The algorithm is similar to typical A* search except for the following differences:

- The search interacts with resource agents in a dynamic fashion.
- The search can be interrupted :
  1. If there are idle resources. A partial schedule (which is called a *committed schedule*) is returned and sent to each manipulator agent. The schedule that has been executed is called the *executed schedule*. The search continues from the committed scheduled.
  2. If one of the resources fails. It checks if the *committed schedule* can be executed. If so, it backtracks to the most promising schedule in which no failed resources have been assigned; it resumes searching from that point. All the executed tasks are excluded from further search.
- Because there will be differences between the planned schedule and the execution results, an execution strategy that maintains the precedence relationships of the planned schedule is used.

The discussion of using of a non-admissible evaluation function to convert the A* search algorithm into an anytime algorithm can be found in [5].

4 Agent-Based Assembly Planning System

We have integrated the anytime scheduling algorithm described in the previous section into an agent-based planning system. In this section, we describe the agent architecture and communication mechanism of our planning system.

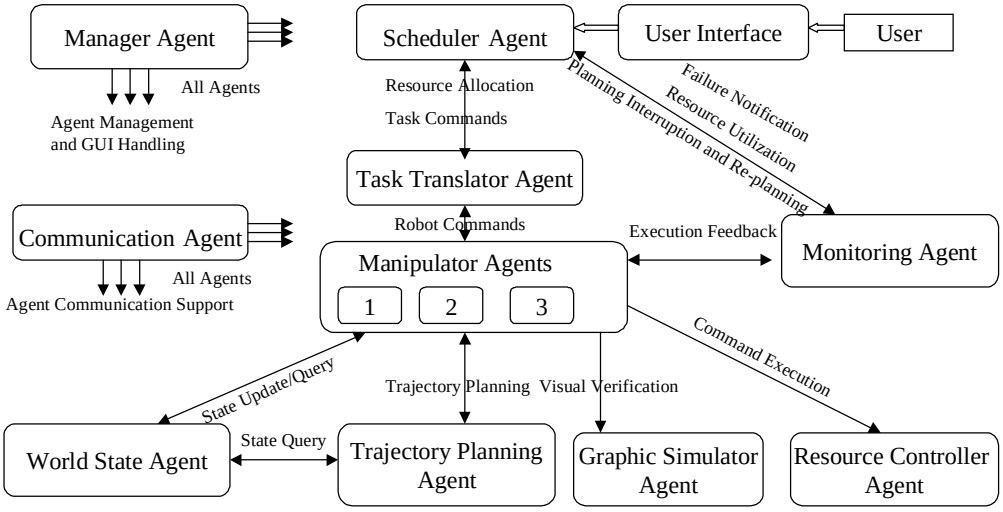


Figure 2: Agent System Architecture

4.1 Agent System Architecture

The performance of an agent-based system depends significantly on how the problem is decomposed into individual agents. As illustrated in Figure 2, we have adopted an agent-architecture based on our previous work [8]. The operation flow of the system is described as follows:

1. The user specifies an assembly task file (with known assembly sequence and precedence constraints) through the user interface.
2. The scheduler agent starts allocating resource and generates a schedule. For each task, the scheduler queries each of the resources for and estimated time of operation (based on the capability, current position, and availability). The scheduler agent returns the best schedule so far whenever it is asked for one by the monitoring agent. It can also re-schedule or abort in case of execution errors or other exceptions.
3. The monitoring agent keeps track of the utilization of the manipulators and interrupts the scheduler agent if there are idle resources. It also reports the results of the execution back to the scheduler.
4. When a resource (manipulator) agent receives a task to be executed, a translation agent converts the high-level task into robot-level commands.
5. Before the resources execute the robot-commands, the trajectory planner generates collision-free trajectories for each resource.
6. Once the trajectory for each resource is available, the user can either send all robot commands to the simulation agent to obtain visualization feedback, or to the controller agent to execute robot commands in our assembly test bed.
7. Each resource agent reports execution errors and exceptions to the monitoring agent.

4.2 Communication Mechanism

For the implementation of our planning system, we use the Task Control architecture (TCA) developed at Carnegie Mellon University [15]. TCA provides a high-level machine independent method for passing messages between agents. TCA is capable of conducting peer-to-peer and broadcast communication. Every agent communicates first with a central server after which the server dispatches the requests to target agents. The agents in the planning system exchange information through messages and requests. The different types of messages are illustrated in Figure 2.

4.3 Agents

We have designed the following agents in our planning system:

- **Communication Agent:** handles agent communication based on the Task Control Architecture (TCA) message-passing mechanism.
- **Manager Agent:** manages the agent society. It is also in charge of the interaction with the user.
- **Scheduler Agent:** generates dynamic schedules.
- **Resource Agent:** provides manipulator information (current position, current state, and capacity). Currently, the only resource agents are the manipulator agents.
- **World State Agent:** provides the geometric information of the manipulators, workspace, and parts. It also contains connectivity relationships between parts and manipulators.
- **Monitoring Agent:** keeps track of task execution results and coordinates scheduling and execution. It also handles exceptions such as manipulator failures.
- **Task Translator Agent** converts high-level task commands into lower-level robot commands. We use motion scripts that contain motion strategies to map high-level task commands into robot commands, for example:

```
# Motion Script for a Pick-and-Place task
MoveRobot $RobotName $PartInitConfig
RobotOpenGripper $RobotName
RobotCloseGripper $RobotName
MoveRobot $RobotName $PartFinalConfig
RobotOpenGripper $RobotName
MoveRobot $RobotName $RobotInitConfig
```

- **Trajectory Planing Agent:** generates collision free trajectories for the manipulators. An online hill-climbing algorithm considering multiple moving objects has been developed. Our current implementation performs the collision detection in 2D. As illustrated in Figure 3, bounding

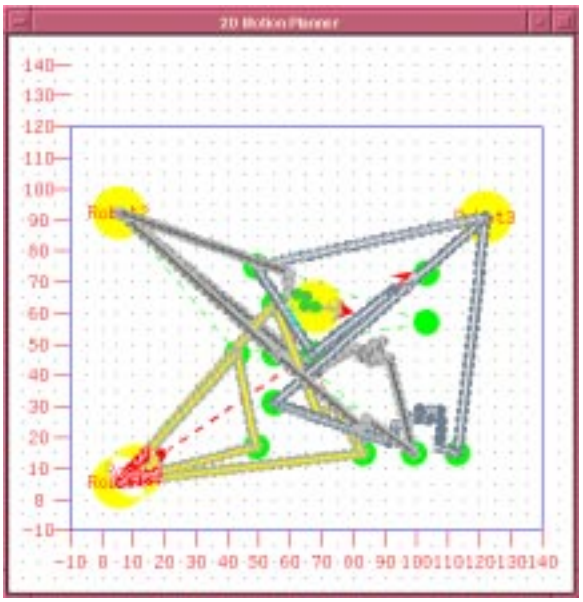


Figure 3: Trajectory Planning

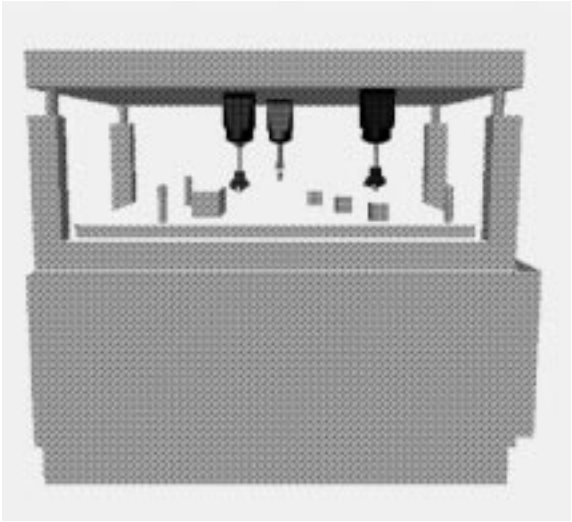


Figure 4: Graphic Simulator

cylinders are used to model the manipulators.

- **Resource Controller Agent:** controls the actual resources in the assembly cell. Ideally there will be one controller for every resource. In our case (RobotWorld), only one controller controls all manipulators.
- **Graphical Simulation Agent:** renders the assembly system using OpenInventor as is shown in Figure 4. Its main purpose is to facilitate debugging and algorithm development.

4.4 Assembly Test Bed

We are currently developing the agent-based planning system for Robot World, a flexible robotic assembly cell developed by Automatix, Inc. As shown in Figure 5, RobotWorld consists of four manipulators, three of which can used to perform assembly tasks (each with 4DOFs). The fourth manipulator is equipped with a camera for calibration and visual feedback. A single controller controls all four manipulators; the other agents running on an SGI communicate with the controller through a serial connection.

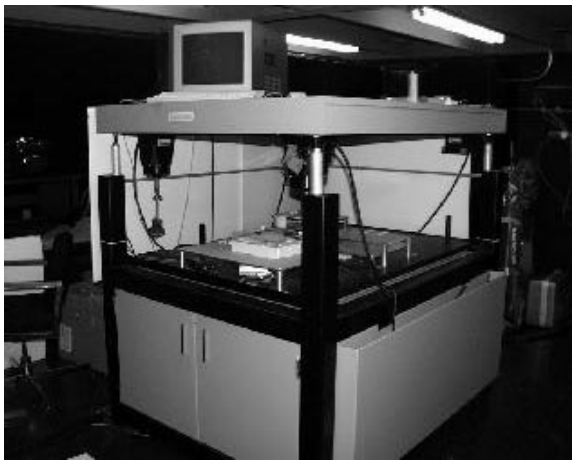


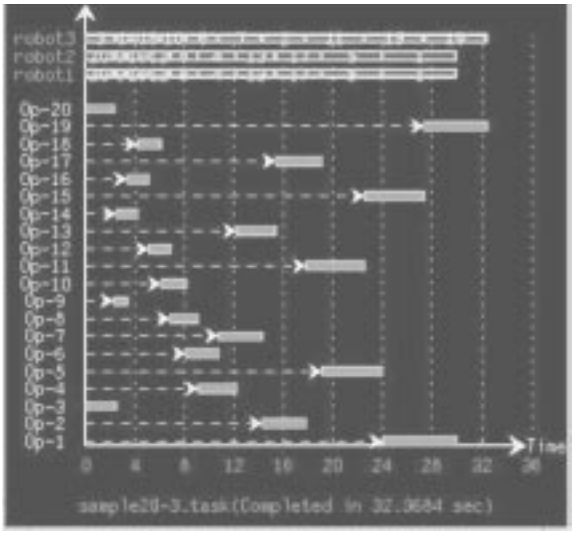
Figure 5: RobotWorld Assembly Cell

5 Simulation Results

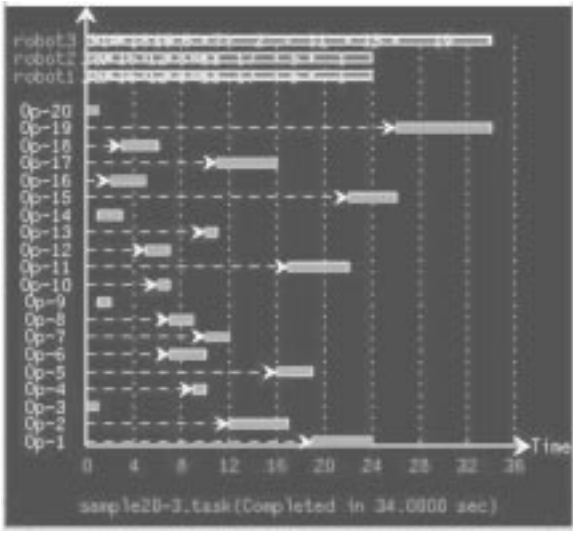
We have tested our on-line planning system using simulated execution results. The actual execution time of an assembly task is simulated by the estimated operation time plus or minus some perturbation. The simulated robots have a 5% failure rate. The results show that our planning system can handle these dynamic changes well.

Figure 6 shows the planned and simulated execution schedules for an assembly job with twenty assembly tasks. Three manipulators are used in this experiment. The precedence relations of the planning result are maintained to handle the discrepancy between planning and execution in simulated execution schedule.

Figure 7 shows the recomputed schedule when one of the manipulators (robot 3) failed.



planned schedule



simulated execution schedule

Figure 6: Planned and Simulated Execution Schedules

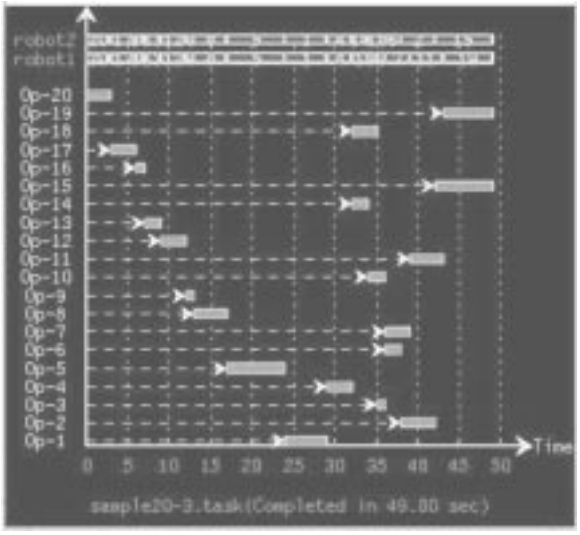


Figure 7: Planned Schedule with One Broken Manipulator (robot 3)

6 Conclusion and Future Work

In this paper, we presented an agent-based on-line planning approach for flexible assembly systems. We focused on assembly task allocation. An anytime algorithm based on A* search has been implemented as part of an agent-based system architecture. We have tested our planning system on several assembly tasks in which unexpected delays and system faults have been introduced. The experimental results show that our approach is adequate for flexible robotic assembly.

Currently we are working on integrating our planning system with assembly sequence generation (IAMS) and our assembly test bed (RobotWorld). The approach is general enough to be applied and extended to other flexible manufacturing systems.

7 Acknowledgements

This research has been funded in part by DARPA under contract ONR #N00014-96-1-0854 and by the Institute for Complex Engineered Systems at Carnegie Mellon University.

8 References

[1] K.R. Baker, *Introduction to sequencing and scheduling*, Weily, New York, 1974.

[2] J.S. Basran, E.M. Petriu, and D.C. Petriu. “Flexible agent-based robotic assembly cell,” *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pp. 3461-3466, 1997.

[3] M. Boddy and T.L. Dean, “Solving time-dependent planning problems,” *Proceedings of the Eleventh International joint Conference on Artificial Intelligence*, pp. 979-984, Detroit, Michigan, 1989.

[4] S.K. Gupta, C.J.J. Paredis, R. Sinha, C.H. Wang and P.F. Brown, "An Intelligent Environment for Simulating Mechanical Assembly Operations," *ASME Design for Manufacturing Conference*, Atlanta, GA, September 1998.

[5] E.A. Hansen, S. Zilberstein, and V.A. Danilchenko, “Anytime Heuristic Search: First Results,” CMPSCI Technical Report 97-50, Computer Science Department, University of Massachusetts, Amherst, September, 1997.

[6] P.E. Hart, N.J. Nilsson and B. Raphale, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2), pp.100-107, 1968.

[7] L. Homem de Mello and A. Sanderson, “A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences,” *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 2, pp.228-240.

[8] J.-C. Fraile, C.H. Wang, C.J.J. Paredis, and P.K. Khosla ”Agent-Based Control and Planning of a Multiple-Manipulator Assembly System,” *Proceedings of the* 1999 *IEEE International Conference on Robotics and Automation, Detroit, MI.1999.*

[9] L.J. Krajewski and L.P. Ritzman, Operations Management: Strategy and Analysis, Addison-Wesley, 1993.

[10] T.C. Lueth, and T. Laengle. “Task description, decomposition and allocation in a distributed autonomous multi-agent robot system,” *Proceedings of the 1994 IEEE International Conference on Robots and Autonomous System*, pp 1516-1523, 1994.

[11] T. Nagata, and J. Hirai. “Distributed planning for assembly tasks by multiple manipulators,” *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp.3522-3529. 1994.

[12] E. Oliveira. “Cooperative multi-agent system for an assembly robotics cell,” *Robotics and Computer Integrated Manufacturing*, Vol. 11, No 4, pp. 311-317, 1994.

[13] D. Ouelhadj, C. Hanachi B. Bouzouia. “Multi-agent system for dynamic scheduling and control in manufacturing cell,” *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, pp. 2128-2133, 1998.

[14] A.A. Rizzi, J. Gowdy, and R.L. Hollis, “Agile Assembly Architecture: An Agent Based Approach to Modular Precision Assembly Systems,” *Proceedings of the 1997 IEEE International Conference on Robots and Autonomous System*, Albuquerque, NM, April 20-25, 1997.

[15] R. Simmons, “Structured Control for Autonomous Robots,” *IEEE Transactions on Robotics and Automation*, 10:1, February 1994

[16] S.V. Vrbsky, J.W.S. Liu, and K.P. Smith, An Object-Oriented Query Processor That Returns Monotonically improving Approximate Answers. Technical Report, UIUCDCS-R-90-1568, University of Illinois at Urbana-Champaign, 1990.

[17] R.H. Wilson, *on Geometric Assembly Planning*. Ph.D. Thesis, Dept. of Computer Science, Stanford University, Technical Report STAN-CS-92-1416, 1992.

[18] S.-Y. D. Wu, and R.A. Wysk, “An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing,” *International Journal of Production Research*, Vol. 27, No. 9, pp.1603-1623., 1989.

[19] S. Zilberstein and S.J. Russel, “Anytime Sensing, Planning and Action: A Practical Model for Robot Control,” *Proceedings of the Thirteenth International joint Conference on Artificial Intelligence*, Chambery, France, 1993.