Architecture for Industry 4.0-based Manufacturing Systems

Achal Arvind

CMU-RI-TR-16-43

July 2016

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Stephen Smith
Zachary Rubinstein
David Bourne
Chung-Yao Chuang

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Keywords: Industry 4.0, Cyber-Physical Systems, Manufacturing, Architecture

For my family and friends

Abstract

Industry 4.0 or the fourth industrial revolution refers to the trend of integrating cyber physical systems into manufacturing. Outfitting manufacturing units with an array of sensors/end effectors provides the ability to build a virtual model of the unit. This virtual model can in turn be used for automated decision making leading to improvement in performance.

We propose a next-generation computational framework for dynamic, data-driven optimisation of production. In keeping with the principles of Industry 4.0, our architecture is easy to reconfigure and connect to various hardware/software devices allowing for quick reconfiguration of factories. It has a live representation of the real world enabling mangers and other users to keep track of activities in the factory. It also provides users with statistics and other abstraction tools to augment decision making capabilities and has the ability to automatically allocate resources thereby allowing for nearly autonomous operation of the factory.

We further demonstrate an implementation of the proposed architecture on a model problem developed in conjunction with Foxconn. The implementation consists of a ROS [7] based event simulator, Rviz based visualization platform and multiple planners. We implement a baseline strategy inspired by the working of Foxconn factories and then show how our architecture improves upon the performance of the baseline method. The proposed architecture yields an almost 100% improvement over the baseline strategy.

Acknowledgments

I would first like to thank my advisers Prof. Stephen Smith and Dr. Zachary Rubinstein who have always been around for me.

I would also like to thank the members on my committee Dr David Bourne and Chung-Yao Chuang for their invaluable feedback.

Finally, I must express my very profound gratitude to my parents and to my friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Author

[Achal Arvind]

Contents

1 Introduction **1**

2 Literature survey **3**

3 Three Layer Architecture **5**
3.1 Interface 6
3.2 Information Processing 6
3.3 Intelligence 6

4 Case Study: Foxconn **7**
4.1 Manufacturing Line Stages 8
4.2 Automated Guided Vehicles 9

5 Interface **11**

6 Information Processing **15**

7 Intelligence **17**
7.1 Baseline 19
7.2 Nearest in Context 21
7.3 Late commitment 23

8 Results **27**

9 Future work **31**

10 Conclusions **33**

Bibliography **35**

List of Figures

1.1 The Four Industrial Revolutions 1

3.1 System Architecture 5

4.1 Sample Foxconn Factory 7
4.2 Stages in the Foxconn Reference Problem 8

5.1 Configuration elements in the simulator 11
5.2 Sample operation in simulator 14

6.1 Manager Information Panel showing the Virtual Factory 15
6.2 Zoomed in view of the visualization 16

7.1 Grouped machines 17
7.2 Problems with the Baseline Strategy 20
7.3 Nearest-in-Context heuristic 21
7.4 Nearest-in-Context example 23
7.5 Late commitment heuristic 24

8.1 Throughput of manufacturing Line 27
8.2 Raw Material in manufacturing Line 27
8.3 Performance of system with stochasticity in process times 28
8.4 Performance of system with failed parts 29

Chapter 1

Introduction

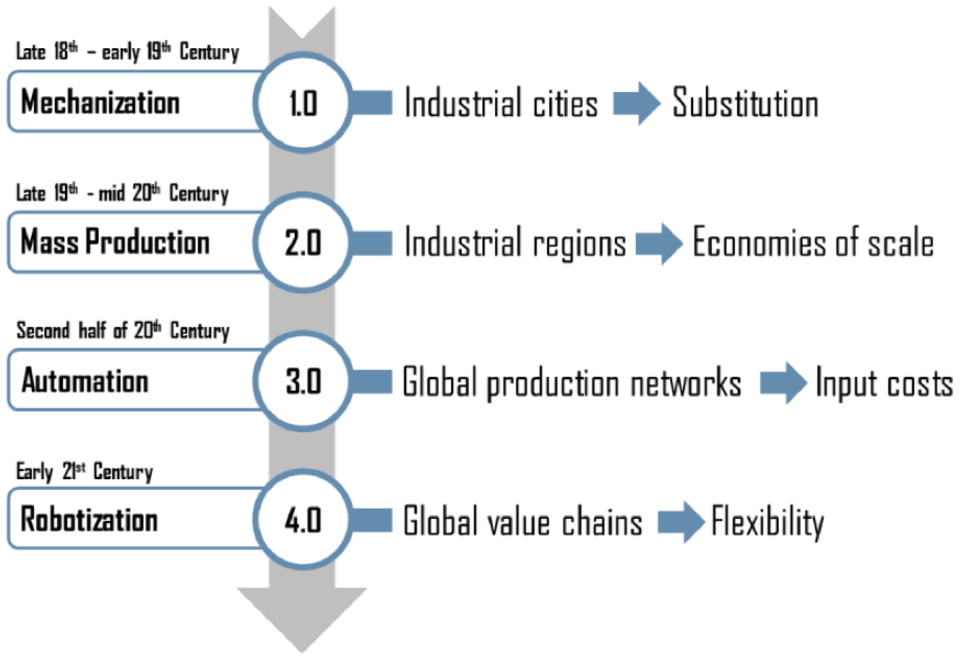


Figure 1.1: The Four Industrial Revolutions

There have been three industrial revolutions up to this point in time and the fourth is in progress. The first revolution started in the late 18th century and was brought about by the advent of the steam engine and focused on mechanization. The second industrial revolution began around the late 19th century and focused on mass production with the introduction of manufacturing lines. This enabled industry to scale up manufacturing with better coordination between labor and machines. The third revolution began around the end of the 20th century with the introduction of the microcomputer. This allowed for the automation of several manufacturing processes using machines that could repeat a series of tasks with high precision with minimal supervision under strictly defined conditions.

The fourth industrial revolution is just beginning to unfold and is based on cyber physical systems [5] [2] which will enable manufacturing units to have higher flexibility in terms of manufacturing processes, the customization of the products and the scale and scope of output.

This research develops and demonstrates a next-generation computational framework for dynamic, data-driven optimization of production. In keeping with the principles of Industry 4.0, the framework

- Allows for interoperability between various platforms and humans by providing a common framework that encodes process models. This allows for more structured conversation in the manufacturing organization.
- Provides information transparency through a live representation of the manufacturing processes that is created by fusing together information from sensors on the factory floor.
- Provides technical assistance to human decision making by providing them a live representation of the factory floor and also through he use of a learning component that will allow the factory to an extent to predict failures and suggest workarounds. It should be noted that the learning component is developed in this work. The framework developed provides an API for a learning component to be inserted and this is further discussed in future work.
- Has the ability to make decisions automatically though the use of a planning component that at the factory floor level coordinates resources to optimize production and keep execution going when anomalous events are detected.

This framework provides several tangible benefits for the manufacturing organization. The core representation of manufacturing processes and resources that is developed enables establishment of a corporate manufacturing knowledge base, promoting formalization of production process alternatives and contingencies which in turn can lead to more informed execution-time decision-making. The live representation of current and projected factory operations enables early detection of production problems and provides a structured framework for making workflow decisions (even if decisions continue to be made manually). The dynamic scheduling middleware utilizes this virtual representation of the current and projected factory state to generate optimized part movement plans and resource allocation decisions (which can also be formulated as decision options and lead to a more efficient manual decision-making process). Ultimately, greater standardization and automation of factory floor decision-making pushes decision support to higher level coordination decisions with supply network partners.

The developed infrastructure is demonstrated and evaluated on a representative problem determined in conjunction with Foxconn. Evaluation is focused on developing a virtual model of the reference problem domain and quantifying the advantage provided by the infrastructure's dynamic scheduling capabilities over current baseline strategies.

Chapter 2

Literature survey

Industry 4.0 describes the increased integration of cyber physical systems in to production and manufacturing. VDMA, Bitkom and ZVEI, three leading German associations defined the goal of industry 4.0 as optimization of value chains by implementing an autonomously controlled and dynamic production.

Due to the specialized nature of the work presented it is hard to make direct comparisons to the work that has already been presented in the field. Several architectures for integrating cyber physical systems have been proposed in the past.

Kolberd et al[4] present an architecture for lean manufacturing using Industry 4.0 They show how Industry 4.0 can be integrated into existing lean production systems and claim that lean production is a natural precursor to Industry 4.0 However while they provide many instances of how CPS and Industry 4.0 can be used in lean production systems, their work does not provide a clear framework for the implementation of a cyber physical system based manufacturing system.

Lee et al[6] proposed the 5C architecture which is a 5 layer pyramidal architecture with layers for Smart Connection, Data to Information Conversion, Cyber Integration, Cognition and Configuration. They also provide a clear description of how to implement their framework. However, to the best knowledge of the authors of this paper, the 5C architecture has not yet been implemented.

Also, it should be noted that there are several instances of Cyber Physical Systems used for material transportation in Industry like Kiva Systems [3], which uses a fleet of autonomous robots to move bins containing material in warehouses; Fetch Robotics [1], which has a system that performs order fulfillment by getting robots to follow humans around the warehouse and Seegrid Robotics which converts forklifts into autonomous agents that move material around large warehouses etc. However, due to the proprietary nature of those systems, the authors of this work have not been able to compare the performance of the proposed system against that of the above mentioned systems.

Current work consists of frameworks that have been proposed but not implemented or implementations in industry with no information of how to go about implementing such a system. The work presented in this paper aims to fill this gap by proposing a simple architecture that will allow for the integration of cyber physical systems into manufacturing and demonstrating that architecture on a representative manufacturing problem.

Chapter 3

Three Layer Architecture

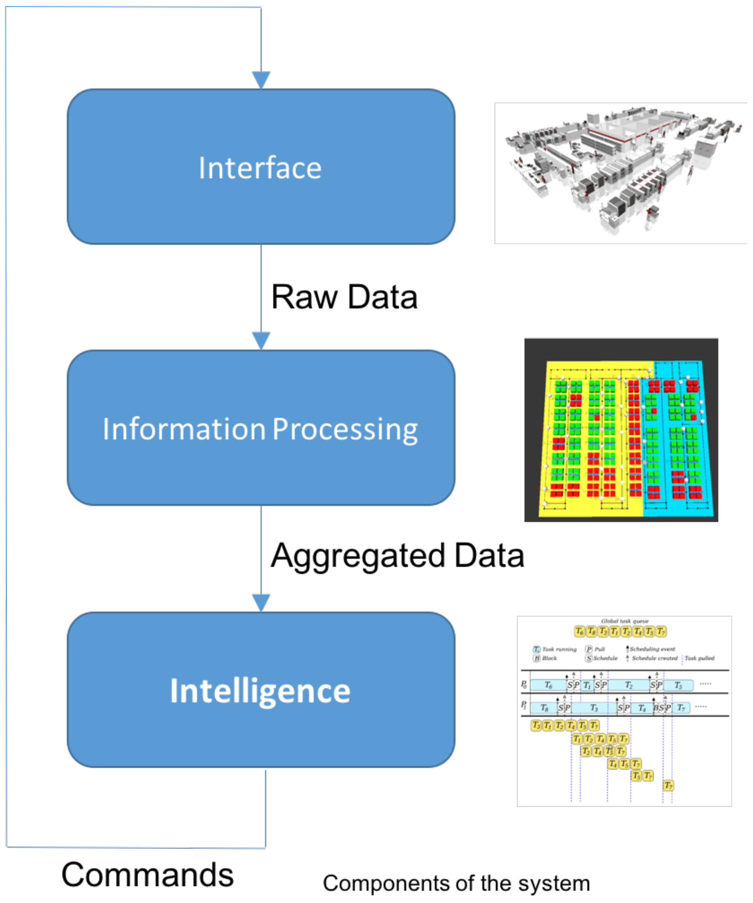


Figure 3.1: System Architecture

Any cyber physical system needs to have modules for interfacing to the physical world, aggregating data, connecting to the cyber computational space and for coordination and intelligence. With this in mind, the three layer architecture has layers for

- Interfacing to the physical world (Information Gathering Layer)

- Data aggregation and connection to the cyber computational space (information Processing)
- Coordination of resources (Intelligence)

Keeping in mind that this architecture is for manufacturing, the layers also have to ensure they follow the design principles of Industry 4.0. The following sections describe the function of each layer and also show how they can be used to attain the tenets of Industry 4.0

3.1 Interface

This layer acts as an interface to the physical world. It serves two functions; Collect information from the all sensors in the factory and relay commands from higher layers to the physical actuators in the factory. One of the key advantages of Industry 4.0 is the ability to have real time information from the factory floor. Therefore, it is imperative to have enough sensors in all the critical systems and the ability to efficiently transfer that information to the Information processing layer. This layer should have the ability to interface to multiple systems easily thereby allowing for interoperability.

3.2 Information Processing

This layer is responsible for aggregating the information collected by the Information Gathering Stage and creating a Virtual Model of the physical world. This virtual model can then be used in two ways: a) The information is sent to a cyber computational space and Machine Learning or similar statistical methods can then be used to create predictive models of the elements in the factory. The data can also be analyzed to create more accurate models of the various elements of the factory which can then be used to tune the overall model of the factory used by the Intelligence Layer. b) The information can also be used to create a real time visualization of the system thereby providing humans valuable insight into the system to enable better decision making. This visualization can also be used to identify anomalous conditions that the intelligence layer misses thereby providing feedback to the system. The virtual model synchronizes with the physical factory floor and this provides the user with a live representation of the factory thus satisfying the information transparency design principle of Industry 4.0 This live model with the statistical analysis provide technical assistance which is another design principles of Industry 4.0

3.3 Intelligence

This layer is responsible for the coordination of all the resources in the factory. It takes as input a model of the factory and the aggregated sensor information and performs the task of resource allocation. This layer provides the system the capability of automated decision making and can manage the manufacturing unit to a large extent without any external supervision. It outputs commands to the various systems in the manufacturing unit that are forwarded to the information collection layer.

Chapter 4

Case Study: Foxconn

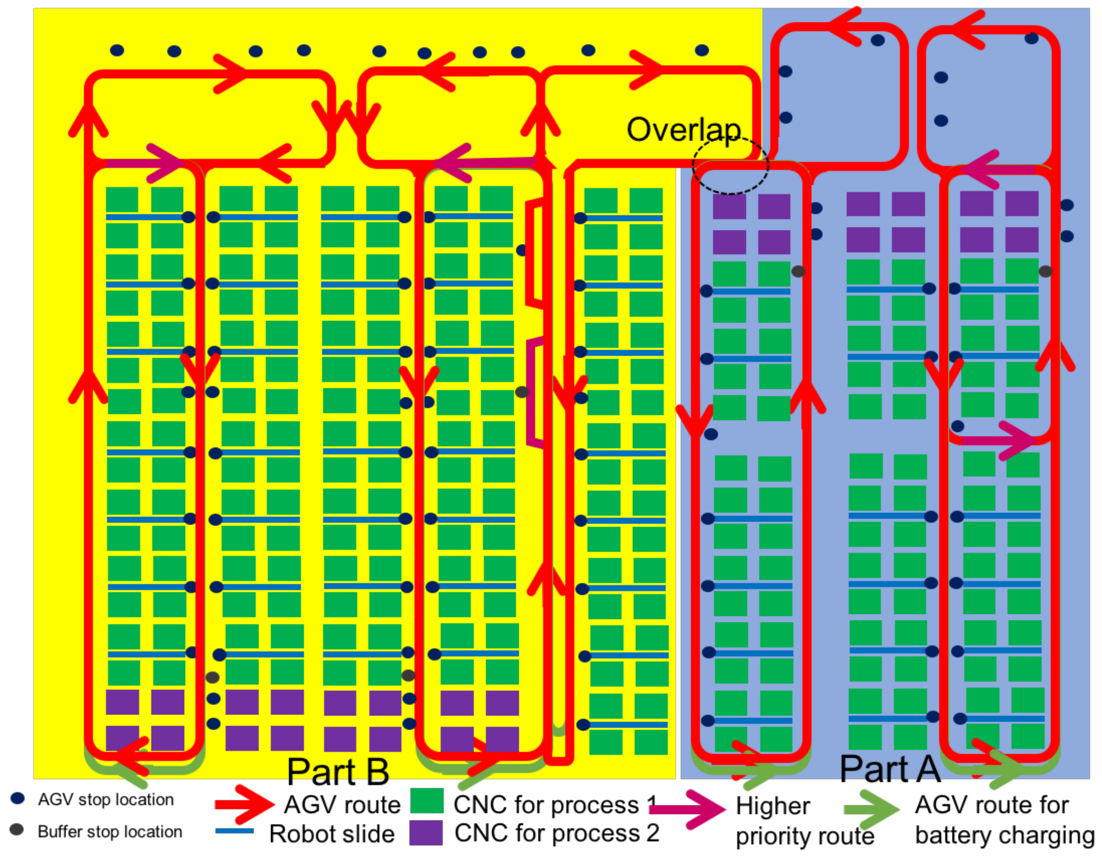


Figure 4.1: Sample Foxconn Factory

The Foxconn problem is in essence a material handling problem. Material needs to be transported from a raw material depot to the two CNC stages (in order) and then finally to the finished material depot. Figure 4.1 shows a graph representing a typical factory with the edges representing the paths that the AGVs (Automated Guided Vehicles) follow. The paths represent magnetic tape on the floor that the AGVs can follow and are strictly unidirectional. Each AGV has the ability to transport multiple trays of parts. Each tray is designed to hold 4 parts. The green and purple squares represent the first and second stages of the machining process respectively.

The following section details the various sections of the manufacturing line and the interactions between them.

4.1 Manufacturing Line Stages

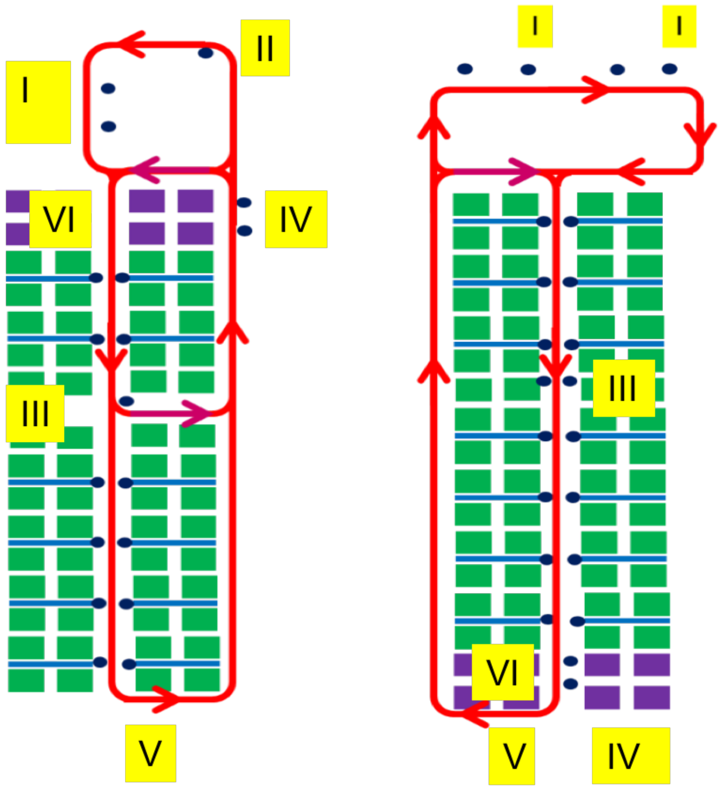


Figure 4.2: Stages in the Foxconn Reference Problem

I **Raw Material Loading Station:** The AGVs pickup raw material at this location. They can pick up to 7 trays at a time. It is assumed for the purpose of this case study that the raw material pickup station has an infinite capacity. A typical loading operation takes around 75 seconds

II **CNC Stage 1:** The stage of the machining operation is completely automated. AGVs load one tray at a time into a CNC group. A robot arm which is part of the CNC group and has end effectors that can grip the raw material and the machined part transfers the parts on the tray into the machine and the finished parts back into the tray. This operation happens in two stages:

(a) First the robot arm transfers all parts from the tray into a buffer on the arm if the buffer on the arm is empty or if the buffer has finished parts in it, then it swaps the finished parts with the raw material.

(b) Once the parts have been transferred into the buffer on the arm, the arm then moves to each CNC machine and swaps the parts on the arm with the parts on the machine if the machines are already loaded or loads the machines if the machines are empty.

This implies that the first stage of the machining process can accept a single tray of parts at a time and has a total capacity of three trays since it has three buffers.

III **CNC Stage 2** The machines in this stage do not have mechanisms to automatically load and unload trays. The AGV drops a pile of trays at this station and humans load and unload parts into these machines. Once a pile of trays have been machined, AGVs pickup these trays.

IV **Finished Material Dropoff** Material that has been machined by the second stage of CNC machines are dropped off at this location. For the purpose of this case study, it is assumed that the finished material drop-off point has an infinite capacity.

V **Battery Swap Station** This station allows the robots to replenish their energy source. In the interest of time, batteries are swapped instead of recharged. Once the depleted batteries are removed from the robots, they are charged and kept aside to be put back into another robot when it arrives at the station.

VI **Raw Material Intermediate Buffer** This buffer is located just before the second CNC stage. There are three types of parts in the manufacturing unit and the AGVs can carry only two piles of trays. Therefore, it is sometimes necessary for the AGV to dump a pile of parts to service a task and avoid deadlocks. This location in the manufacturing line facilitates this dumping of material.

4.2 Automated Guided Vehicles

The AGVs used in this setting are designed to run on tracks and move only in one direction. They have electric drive-trains and have a mechanism to swap out the battery. They also have two types of tray transfer mechanisms. One type is used to transfer a single tray into an automated CNC stage and the other type of mechanism is used to transfer the entire pile of trays. Each AGV can carry up to two piles of trays at a time and each pile has a maximum capacity of 7 trays.

The following chapters describe how the Three Layer Architecture was implemented for the Foxconn case study.

Chapter 5

Interface

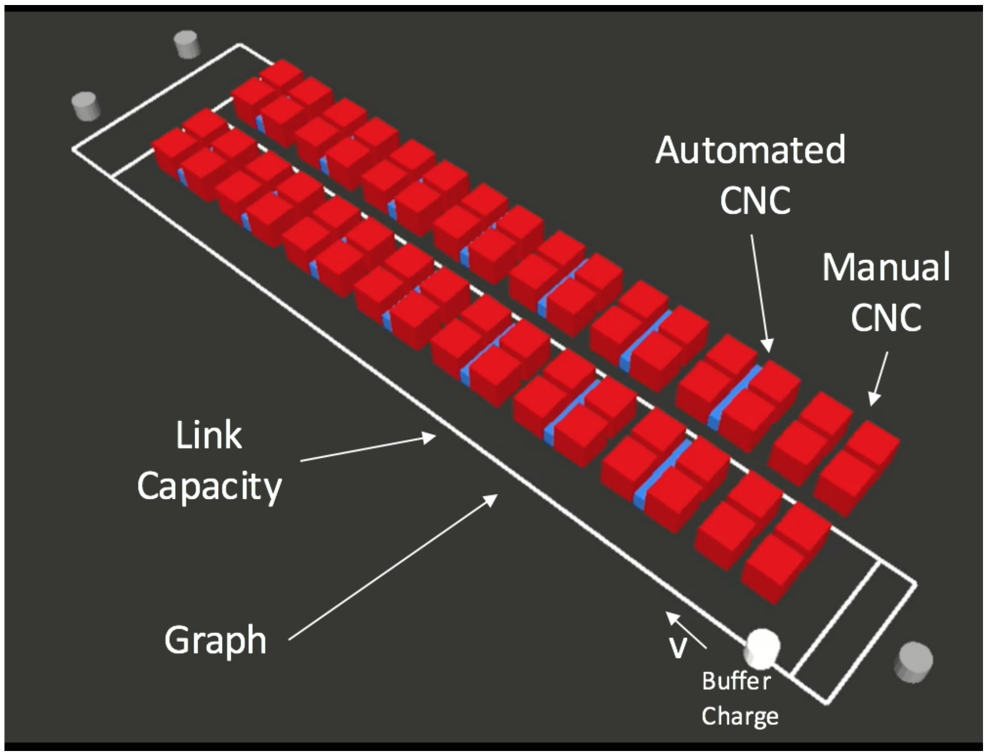


Figure 5.1: Configuration elements in the simulator

This layer acts as an interface between the physical world and the higher levels of the three layer architecture. To allow for the framework to be implemented and the bugs worked out, it is preferable to first connect the framework to a simulator. The section below details the need to build a custom simulator as well as the features of the said simulator.

Most current simulators fall into two categories:

1. **Simulators that model Physics:** These are simulators like Gazebo or Stage that mainly model the physics of the robot and provide the ability to access sensors on the robot. While these simulators accurately model the environment and get close to operating the robots in

the real world, due to the nature of the computations, they don't scale very well with the number of robots or physical constraints.

2. **Process Flow Simulators:** These simulators are great for laying out the floor of the factory and performing optimization routines but they are not very good at simulating interactions with the physical component of the manufacturing system.

What the Foxconn domain really needs is a combination of both these kinds of simulators that has the following features:

1. **Lightweight:** To ensure that it scales well for large manufacturing lines.
2. **Event Based:** The main focus of the simulator should be macro events and the intricate dynamics of robot movement etc. In other words, it should focus on the times at which events occur.
3. **Enforce Spatial Constraints:** Should ensure that robot are allowed to move to locations that are feasible.
4. **Easy to configure:** Should be simple to change the various parameters in the simulation.
5. **Data Driven:** The main components of the simulation like the model of the factory, configuration of the robot should be data driven and should be read in through configuration files.

With these requirements, a simulator was built with ROS as the communication platform to facilitate interaction between the various components in the factory. To emulate a true internet of things system, each component in the system functions as a separate entity and communicates with each other through the passing of messages and the invocation of services. The events that trigger these messages and services are detailed in the Intelligence layer chapter. To ensure that the simulator was lightweight, only simple robot dynamics were modeled and the interactions between the various components were made asynchronous.

Given that the scheduling domain is concerned with the timing of the operations, it was not required to model the dynamics of the machining operations. The simulator therefore functions by treating each operation as a block of time. Data files contain information about the duration of each operation and simulator transitions into a waiting state until the operation has been completed. This ensures that processing space is not used between the start of the operation and the end of the operation.

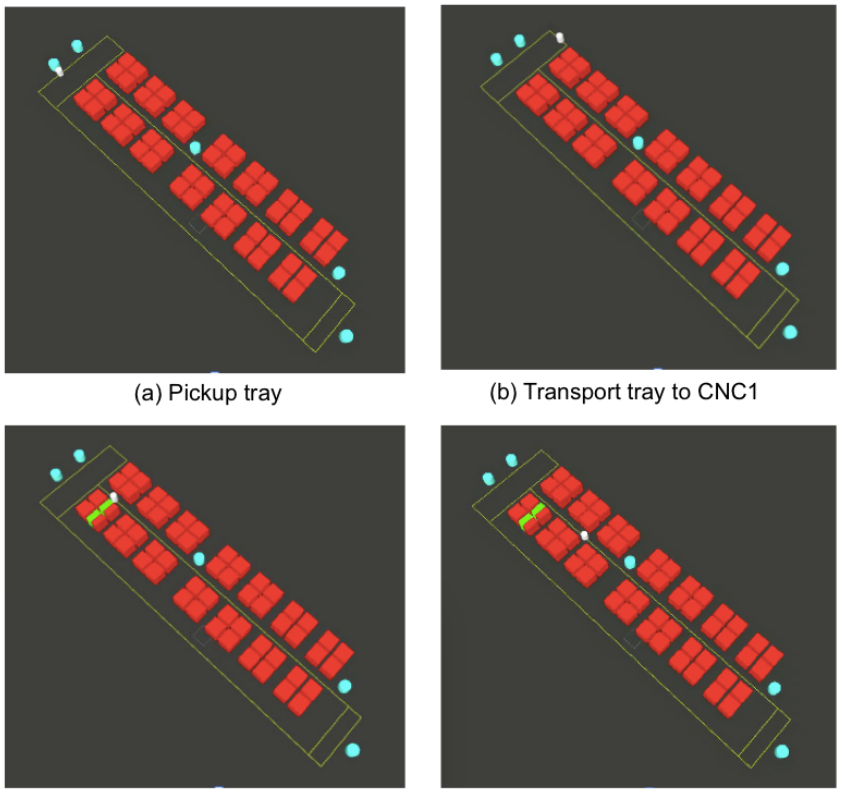
Internally, the simulator models the path of the AGVs as a graph and creates queues for each edge (See Figure 4.1). The queues have different capacities based on the length of the edge. The simulator ensures that the order of robots in the queue is maintained. It also enforces a minimal separation distance and treats each edge and vertex as a critical resource thereby enforcing spatial constraints.

The simulator does not have any information about the layout of the factory or the type/duration of the operations internally. All this information is stored in JSON files that allows for easy reconfiguration of the system. Changing the simulation from one type of factory to another is as simple as changing the input JSON file. For the Foxconn reference problem, the following processes, interactions and constraints were implemented in the simulator:

1. **Processes in the factory**
   (a) Robot Dynamics

        i. Speed
        ii. Buffer
        iii. Battery Level
        iv. Following Model
    (b) Factory Floor Configuration
        i. Stop Locations
        ii. Link Capacity
    (c) CNC Dynamics
        i. Automated arm dynamics
        ii. Buffer
    (d) Process times
        i. CNC Swap
        ii. Battery Swap
        iii. Material loading/unloading

2. **Model Interactions in Factory**
    (a) Robot CNC interaction
    (b) Robot battery swap
    (c) Raw material pickup
    (d) Finished product dropoff
3. **Enforce constraints in factory**
    (a) Spatial
    (b) Robot speed
    (c) Material
    (d) Battery



(a) Pickup tray (b) Transport tray to CNC1

(c) Drop off tray (d) Proceed down track

Figure 5.2: Sample operation in simulator

Figure 5.2 shows a series of screenshots from a simulation run. In this example run, a single AGV (represented as a small white cylindrical base) moves along the track of a single production line, first picking up a raw material pickup station (Fig 5.2.a), then transporting the tray to the first CNC machining process (Fig 5.2.b), unloading the tray at this location (Fig 5.2.c), and moving forward to take on its next task (Fig. 5.2.d). CNC machines announce completion of their tasks, need for more material etc. to their respective managers (software agents that govern each bank of CNC machines; four machines form a bank). These managers, in turn, request service from AGVs when a tray drop-off or pick-up is needed. The allocation of AGVs to these process manager is decided by higher levels of the architecture.

The input to the interface layer is commands for the AGVs and the output is location and state information of the AGVs along with the state of the CNCs, raw material buffer and finished product buffer.

Chapter 6

Information Processing

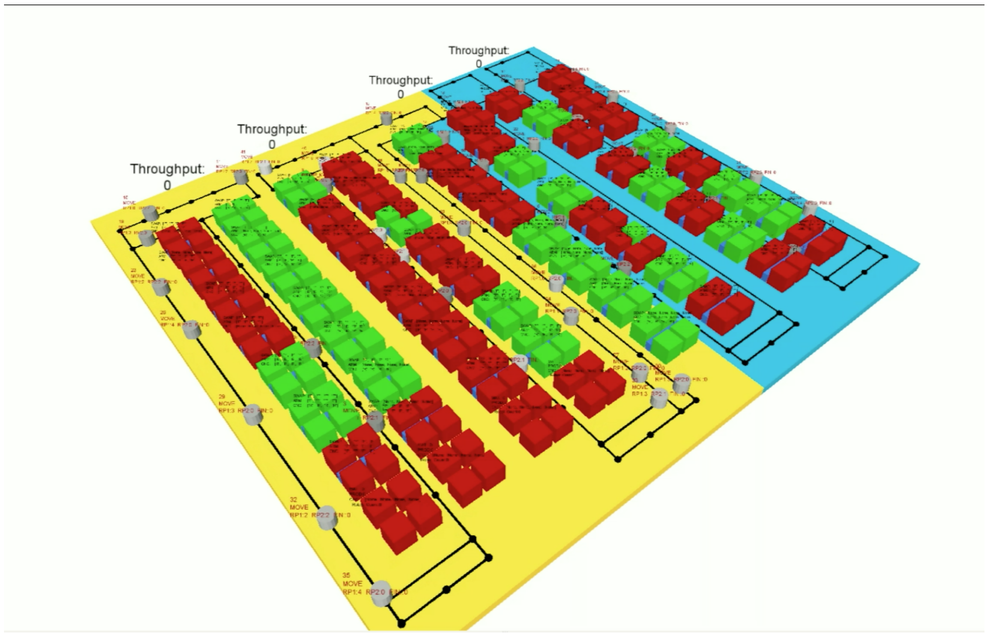


Figure 6.1: Manager Information Panel showing the Virtual Factory

The main responsibility of this layer is the aggregation of information coming in from the information collection layer. It also serves as an interface to the cyber physical computational space and as a logging module.

For the Foxconn scenario, the information processing layer collects the state of the robots (location, charge, buffer levels), CNC machines (state, buffer level), finished product count and count of raw material in manufacturing line (raw material on robots and in the intermediate buffer) from the interface layer.

The collected information is fused together to create a visualization that shows the state of the entire factory in real time. This acts as a *managers control panel* and provides complete information transparency into the real factory. The virtual factory visualization uses color to represent the state of the CNC machines (processing/not processing) and the robot (battery charge level). It also uses a text overlay to provide the viewer with real time information about the state of the various processes.

This layer also logs the aggregated sensor data. The logs are used to compute statistics to measure system performance. This data can also be used to refine the model for further simulations.

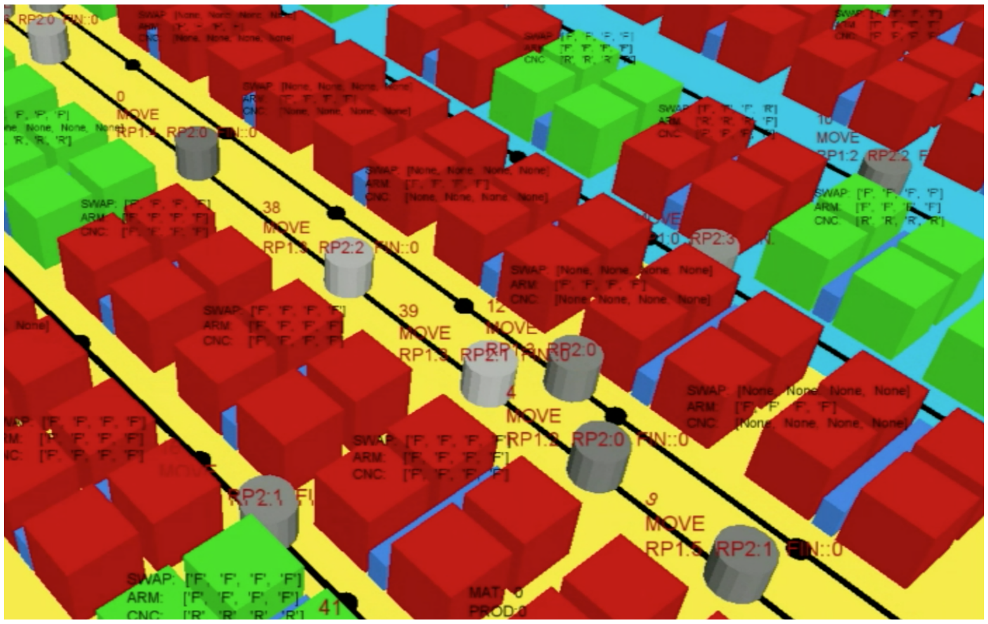


Figure 6.2: Zoomed in view of the visualization

A server collects information from every agent in the system to create a visualization of the Foxconn AGV scheduling problem. This visualization uses the RViz platform on ROS to display the virtual factory in real time. The CNC machines turn green when material is being processed and red when they are idle. Figure 6.2 shows a zoomed in view of the elements displayed in the visualization. The floating text on top of the AGVs and CNC machines display the state of the AGVs (AGV ID, current task and buffer) and the state of the CNC machine (Buffer). The AGVs change their color from white to black based on the battery level. This allows for the user to get a clear picture of the state of the system with a glance. Additionally, information like *throughput* is displayed at top of each manufacturing line.

In essence, this layer collects information from the factory and aggregates it to form a cohesive model of the factory and sends it along to the intelligence layer.

Chapter 7

Intelligence

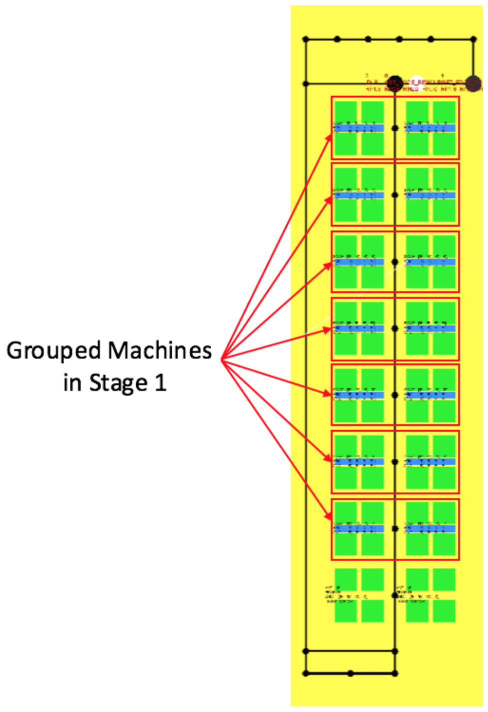


Figure 7.1: Grouped machines

Signal	Significance	Agent(s)
AV	Idle and available to task	AGV
NM	Need raw material	CNC, Left/Right Bank Manager, Machine Manager
S	Have processed material and need raw material	CNC, Left/Right Bank Manager, Machine Manager

Table 7.1: Signals in the system

```
Function RobotCallback(event):
    if event.state.battery>threshold then
        if task_list not empty then
            addToFreeList(event.info.ID)
            call planner
    else
        task event.info.ID to battery swap
    end
    return

Function CNCCallback(event):
    process_ID ← getProcessMapping(event.CNCType)
    heuristic_value ← getHeuristicValue(event.CNCLocation)
    addTask([process_ID, heuristic_value, event.process_type])
    if robot_list not empty then
        call planner
return
```

Algorithm 1: Callbacks for Robot and CNC events

Agent	Responsibility
CNC	Process material in the system
Left Bank Manager	Manage cluster of 4 machines on left of manufacturing line
Right Bank Manager	Manage cluster of 4 machines on right of manufacturing line
Machine Manager	Manage left and right banks managers
AGV	Transport material in the system

Table 7.2: Agents in the system

Before describing the algorithms that manage the intelligence layer in detail, we describe the assumptions we make about how the factory operates and the way that we model it. Each component in the manufacturing facility, e.g., an AGV or a CNC machine, is assumed to have some computational capability and is represented by a software agent. As detailed in the interface layer, agents achieve tasks through invocations of services and coordinate by passing messages to each other. Each component in the manufacturing facility operates independently.

As CNC machines perform tasks and AGVs move material through the factory, they signal events to notify the system of state changes. These events, in turn, trigger allocation decisions. CNC machines signal a **NM** event (Need Material) to indicate that they are idle and require new material and an **S** event (Swap Material Processed) to indicate (1) that they have processed material that is ready to be moved and (2) they require new material. AGVs signal an **AV** event (Available to Task) when they have completed a task to indicate that they are now available to perform a new task. AGVs also send status update messages to the system when the state of their buffers, batteries and locations change.

Each CNC machine processes a single part. To increase pipelining and to reduce complexity, we aggregate the responsibility of coordinating groups of CNC machines into CNC Machine

Manager agents. In our reference problem (see Figure 7.1), which comprises a sequential 2-stage CNC machining process, each CNC machine manager represents the two clusters of 4 CNC machine groups that are serviced (on the left and right) at any given AGV stop location. Each CNC machine manager collects events from its constituent individual CNC machines and, when all 8 individual CNC machines reach the same state (i.e., **NM** or **S**), it signals a single corresponding event to the system. In essence, CNC machine managers abstract its block of 8 constituent CNC machines into one entity. Each time a call is made, 2 trays of parts need to be picked up and/or dropped off. As CNC machine managers signal **NM** and **S** events, the system determines which AGVs to allocate to these requests. Internally, each request is transformed into a service task, represented by the tuple (<Process ID>, <Heuristic value>, <Task ID>). The <Process ID> corresponds to the process stage (i.e., stage 1 or 2 in the reference model) of the requesting CNC machine group. The <heuristic value> is algorithm-dependent and determines which request within a process stage is serviced first. The <Task ID> is the type of service that is being requested, i.e., **NM** (drop off of new material) or **S** (pickup of processed material followed by drop off of new material). Algorithm 1 shows the psuedocode for this process. At any point during execution, there are a set of available AGVs (AGVs that are currently unassigned (and are not engaged in recharging) and a set of unassigned service tasks. In allocating AGVs to services tasks, the strategies specified below assume that unassigned tasks are processed in ascending order with <process ID> as the primary key and the heuristic value as the secondary key.

7.1 Baseline

The baseline allocation algorithm is triggered every time there is a NM or an S event. In the baseline algorithm, events with a stage-2 Process ID are prioritized over those with a stage-1 Process ID. The heuristic value for a service task reflects the position of the requesting CNC cluster relative to the beginning of the manufacturing line. Thus, for service tasks within the same manufacturing stage, AGVs are allocated to service tasks from CNC clusters closest to the beginning of the factory line first. This leads to poor allocation of AGVs in several situations. Below are two such scenarios:

1. Consider the scenario on the left in Figure 7.2. It can be seen that a new task appears just below the location of a robot that is currently executing a task. The planner allocates this task to the only available robot in the system which is just below the location of the task causing that robot to circle around the manufacturing line to get the task.
2. The second scenario occurs when the tasks occur in an order that causes robot to be allocated in such a way that the one robot blocks the path of the other robot. Consider the Consider the scenario on the right in Figure 7.2. When the first task arrives in the system, the nearest robot available is the free robot at the top and therefore, this robot is allocated. Now, when the second task appears in the system, the robot at the bottom is allocated to this task but since the order of the robots is exactly opposite to the order of the tasks, the robot at the bottom has to wait till the robot at the top finishes its task.

```
Function getProcessMapping(cnc_type):
    if cnc_type = 1 then
        process_mapping ← 1
    else
        process_mapping ← 0
    end
return process_mapping

Function getHeuristicValue(cnc_location):
    heuristic_value ← getDistance(manufacturing_start_location, cnc_location)
return heuristic_value

Function Planner():
    if task_list not empty and robot_available_list not empty then
        while task_list not empty or robot_available_list not empty do
            minHeap(task_list)
            top_list ← pop(task_list)
            [robot_id, distance] ← getMinDistance(top_task, robot_available_list)
            removeFromList(task_list, top_task)
            assignTask(robot_id, top_task)
        end
return
```
Algorithm 2: Psuedocode for generating heuristic value and planner for the baseline strategy

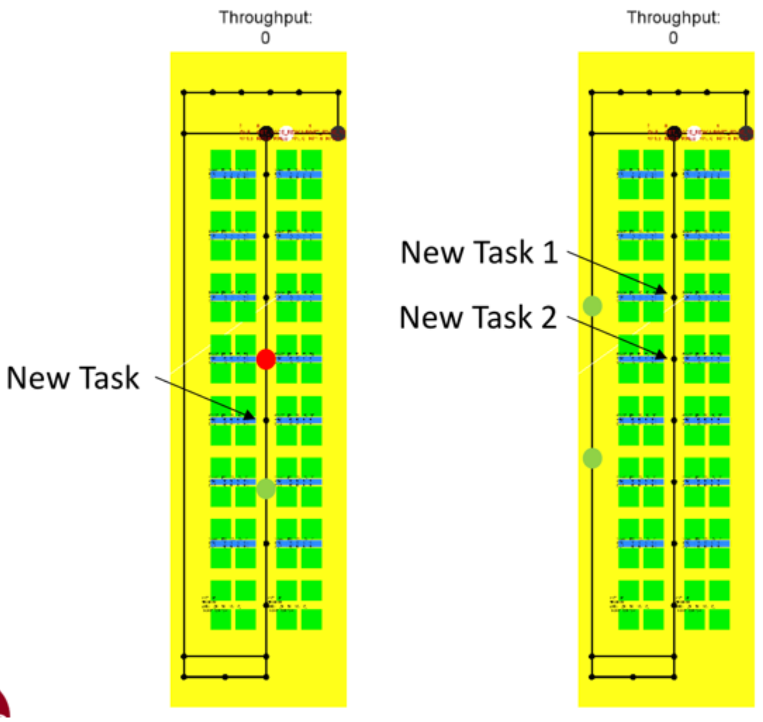
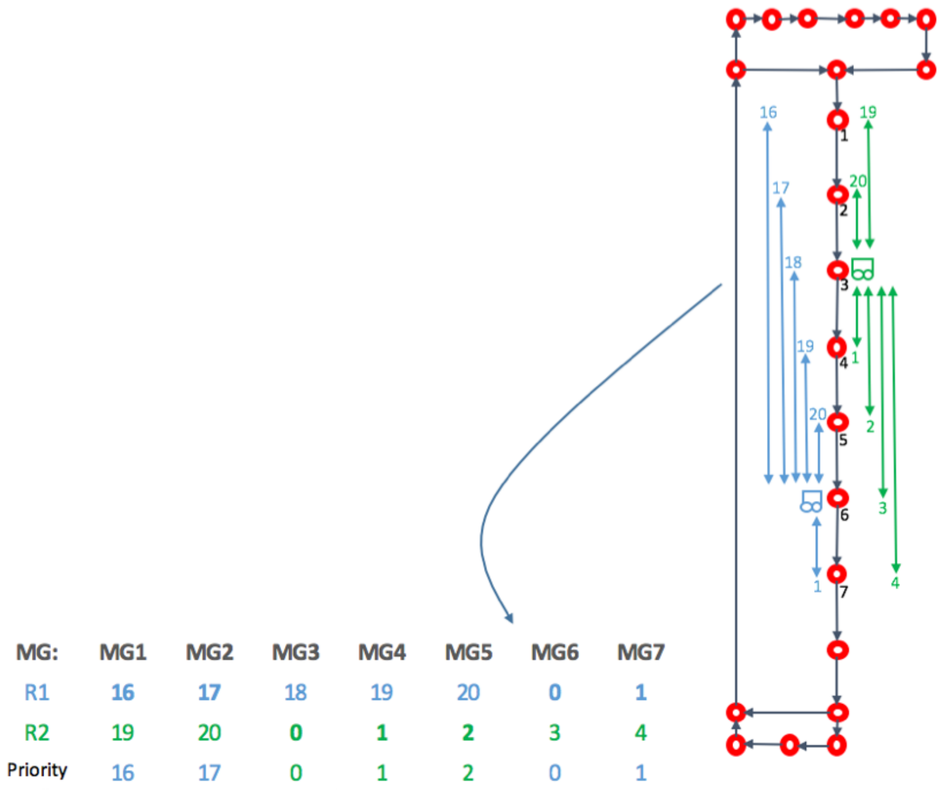


Figure 7.2: Problems with the Baseline Strategy

7.2 Nearest in Context



MG:	MG1	MG2	MG3	MG4	MG5	MG6	MG7
R1	**16**	**17**	18	19	20	**0**	**1**
R2	19	20	**0**	**1**	**2**	3	4
Priority	16	17	0	1	2	0	1

Figure 7.3: Nearest-in-Context heuristic

The nearest-in-context algorithm operates similarly to the baseline algorithm, except that the heuristic value for a service task is the distance to the nearest AGV (as opposed to distance to the beginning of the manufacturing line). This heuristic allocates service tasks to AGVs that are closest to requesting CNC Manager Agent, rather than allocating AGVs based on their proximity to the beginning of the line. Figure 7.3 shows the process of calculating the heuristic value with two AGVs in the system with tasks at each process 1 CNC location.

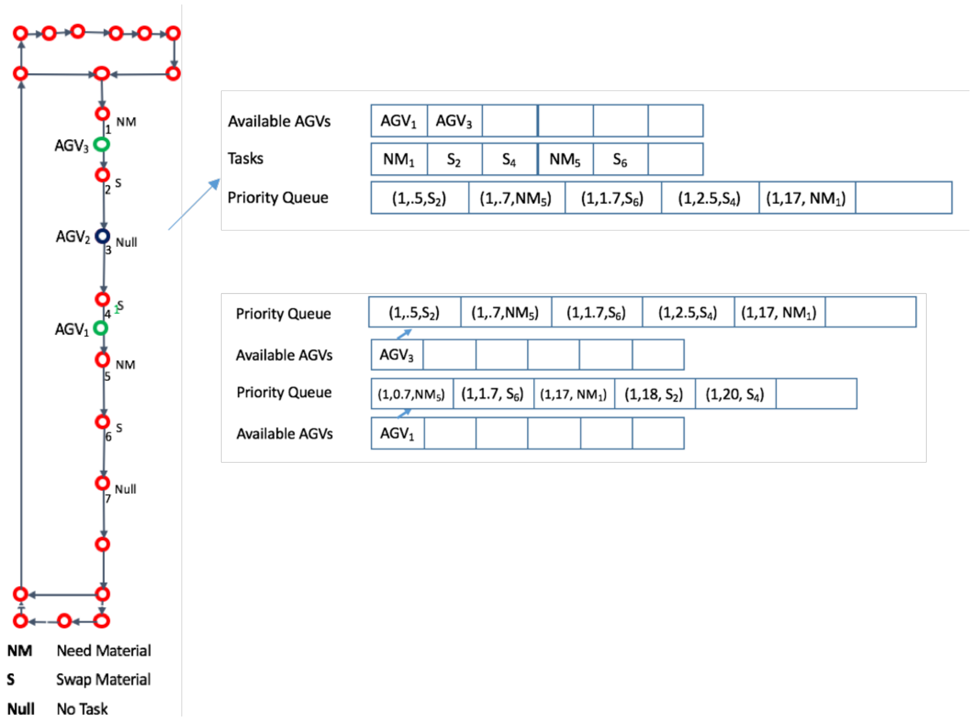
```
Function getProcessMapping(cnc_type):
    if cnc_type = 1 then
        process_mapping ← 1
    else
        process_mapping ← 0
    end
return process_mapping

Function getHeuristicValue(cnc_location):
    for robot in robot_available_list do
        distance_list.append(getDistance(robot.location,cnc_location)
    end
    heuristic_value ← min(distance_list)
return heuristic_value

Function Planner():
    if task_list not empty and robot_available_list not empty then
        while task_list not empty or robot_available_list not empty do
            Recalculate heuristic values
            minHeap(task_list)
            top_list ← pop(task_list)
            [robot_id, distance] ← getMinDistance(top_task, robot_available_list)
            removeFromList(task_list, top_task)
            assignTask(robot_id, top_task)
        end
return
```

Algorithm 3: Psuedocode for generating heuristic value and planner for the nearest in context strategy

The advantage of this new ordering is demonstrated in the following example. Consider the scenario in Figure 7.4, where there are multiple AGVs inside the manufacturing line and positioned at stage 1 CNC machine group locations. In this example, the nearest-in-context algorithm allocates AGV3 to S2 and AGV1 to NM5, which is a significant improvement over the baseline algorithm. This would have allocated AGV1 to NM1 and resulted in AGV1 having to unnecessarily circle around the manufacturing unit. In general, the nearest-in-context algorithm leads to a behavior where, once an AGV has entered stage 1 of the manufacturing line, it can service multiple *stage-1* requests before exiting this portion of the line, which improves the utilization of the AGVs.



Available AGVs	AGV$_1$	AGV$_3$				
Tasks	NM$_1$	S$_2$	S$_4$	NM$_5$	S$_6$	
Priority Queue	(1,.5,S$_2$)	(1,.7,NM$_5$)	(1,1.7,S$_6$)	(1,2.5,S$_4$)	(1,17, NM$_1$)	

Priority Queue	(1,.5,S$_2$)	(1,.7,NM$_5$)	(1,1.7,S$_6$)	(1,2.5,S$_4$)	(1,17, NM$_1$)	
Available AGVs	AGV$_3$					
Priority Queue	(1,0.7,NM$_5$)	(1,1.7, S$_6$)	(1,17, NM$_1$)	(1,18, S$_2$)	(1,20, S$_4$)	
Available AGVs	AGV$_1$					

Figure 7.4: Nearest-in-Context example

It should be noted that while the nearest-in-context planner solves one of the problems encountered by the baseline planner, it still cannot handle the scenario when robots are allocated in the reverse order of the tasks (the second scenario amongst the two scenarios that the baseline planner could not handle)

7.3 Late commitment

The least commitment algorithm assigns tasks to AGVs closer to execution time. In other words, it commits AGVs to tasks later in the process than either the baseline or the nearest-in-context allocation algorithms. The intuition behind the least-commitment algorithm is to ensure that bottleneck resources are kept continuously busy, hence maximizing throughput. In our reference problem, the bottleneck resources are the CNC machine groups that provide stage 1 processing, since stage-1 is the longest duration processing step. The first 7 machine groups in the manufacturing line depicted in Figure 1 are dedicated to stage-1 processing.

To achieve this goal, the least-commitment algorithm has a slightly different control structure than the other algorithms. It operates as follows:

1. Un-tasked AGVs proactively move to the depot, load raw material up to their maximum capacity and then move to a designated wait location at the beginning of the manufacturing line. If the wait location is occupied, the AGVs queue up.
2. Tasks are allocated to AGVs as they arrive at the wait location (delaying commitment until an AGV is loaded and ready to enter the manufacturing line). When an AGV arrives

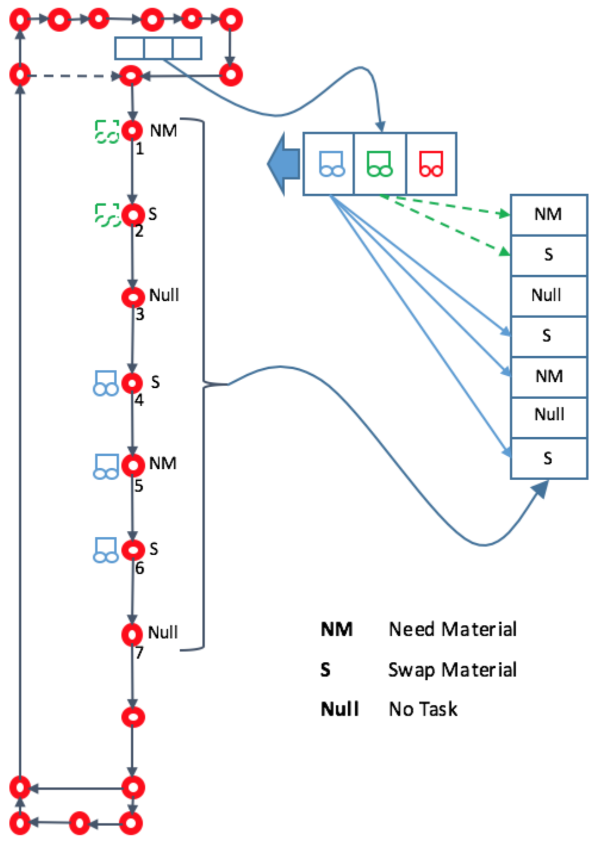


Figure 7.5: Late commitment heuristic

at the wait location, the set of active service tasks for stage 1 are searched to see if a subset of pending tasks can be found that will totally consume the AGVs raw material load (recall that each AGV can simultaneously carry both raw material and processed material in two separate buffers, allowing it to pick up and drop off material at any given stop). In the case where multiple subsets of tasks could be assigned, options are ordered using the same service task tuple representation used by the other algorithms. In this case, the heuristic value is the position of the CNC machine cluster associated with the task (under assumption that CNC machine clusters are numbered in descending order from the beginning of the line to the end). Thus, options (subsets of service tasks) located later in the line are given priority over earlier ones. Only the AGV at the wait location is considered for task assignment. The AGV is released to execute its assigned service tasks only when a set of tasks is found that consumes all of its raw material. Otherwise, the AGV waits for a new a service task to emerge and the search is repeated with the new task list.

3. Upon arrival at the Stage-2 CNC processing step, any processed material that was picked up by the AGV at stage-1 stops is dropped off. If there is any finished material at CNC process 2, then that material is picked up.
4. Upon exiting the machining area, the AGV will drop off any finished product that they

```
Function getProcessMapping(cnc_type):
    if cnc_type = 1 then
        process_mapping ← 1
    else
        process_mapping ← 2
    end
return process_mapping

Function getHeuristicValue(cnc_location):
    heuristic_value ← getDistance(manufacturing_end_location, cnc_location)
return heuristic_value

Function Planner():
    for robot in robot_available_list do
        if robot.location is manufacturing_start_location then
            if len(task_list) ≥ (robot.raw_material_buffer/2) then
                assignTopTasksToRobot(task_list, (robot.raw_material_buffer/2))
        else if robot.location in stage1_stop_locations then
            if robot.stage1_processed_material > 0 or CNC.stage2.finished_material >0
            then
                Task Robot to service stage 2
        else if robot.location in stage2_stop_locations then
            if robot.battery_level < threshold then
                Task robot to swap battery
            if robot.finished_material >0 then
                Task robot to drop finished material at finished material buffer
            if robot.raw_material < threshold then
                Task robot to pick up raw material at raw material buffer
            Task robot to move to manufacturing start location
    end
return
```

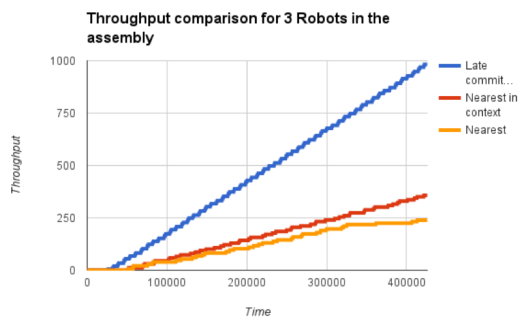
Algorithm 4: Psuedocode for generating heuristic value and planner for the late commitment strategy

are carrying and proceed to the depot to repeat the above process. If the battery charge of an existing AGV is below a threshold, the AGV goes to the battery swap station before proceeding to the final product drop-off and depot.

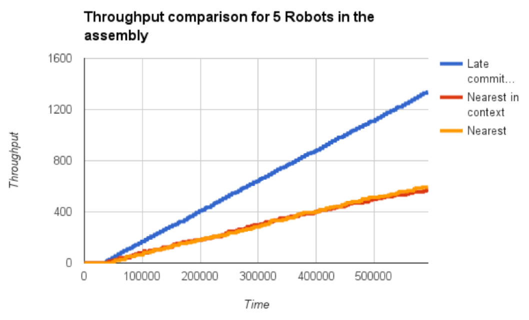
Figure 7.5 shows an example of how the least-commitment algorithm works. There are three AGVs at the wait point and 5 service tasks in the system. Since each AGV can carry 6 trays, each AGV can service 3 tasks at a time. The first AGV is assigned to the tasks at the bottom of the manufacturing line. At this point, there are two pending tasks in the system. Since the second AGV can service 3 tasks, there are not enough tasks in the system to ensure that the raw material in the second AGV is completely consumed. Therefore, the second AGV waits until another request is made by the CNC machines. Note that the maximum capacity of the AGV raw material buffer has been reduced to 6 to ensure that the AGV consumes the material before reaching the end of the first stage of manufacturing.

Chapter 8

Results

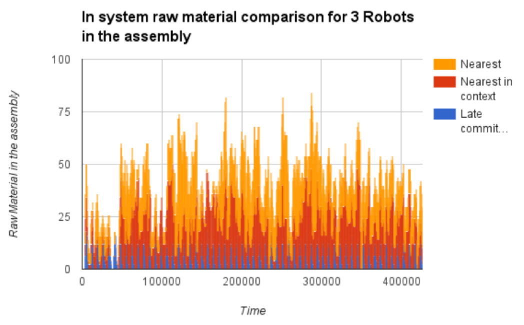


(a) 3 Robots

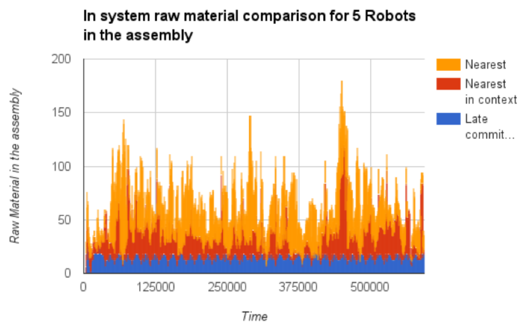


(b) 5 Robots

Figure 8.1: Throughput of manufacturing Line



(a) 3 Robots



(b) 5 Robots

Figure 8.2: Raw Material in manufacturing Line

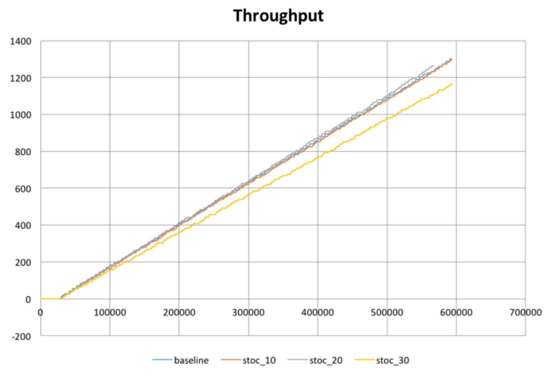
Simulation runs of the reference factory (depicted in Figure 4.1) were performed for two scenarios: 3 AGVs per manufacturing line and 5 AGVs per manufacturing line. Each strategy was run with each scenario and throughput and in-process inventory numbers were collected for each run. Figures 8.1 and 8.2 shows the results. Each simulation was run for 8 hours of factory time and for each allocation strategy.

Two metrics were used for the evaluation; throughput and raw material in the system.

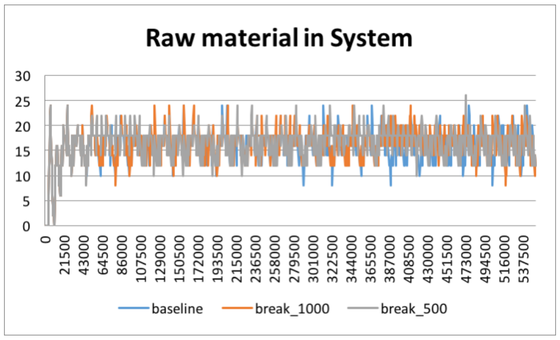
1. **Throughput:** This is a direct measure of performance in the system. It represents the number of parts that are output by the manufacturing line. Higher throughput represents better performance. It can be seen that the Late-Commitment strategy performs significantly better than the Nearest and Nearest-in-Context strategies with an almost 98% improvement. This improvement is due to the fact that late-commitment ensures that the idle time of machines that are part of process 1 is almost close to zero.

2. **Raw material in the system:** While this metric doesnt directly measure the output of the system, it can be seen as a metric that measures the efficiency with which the allocation strategy handles raw material. It can be seen that while Nearest and Nearest in Context are highly unstable and retain a lot of raw material in the system, late commitment has almost no raw material in the system and is stable. This is because late commitment ensures that no robot leaves the first CNC process with any raw material in the buffer. This way the only raw material that is in the manufacturing line is the material that the robots are carrying. On the other hand, the nearest and nearest in context strategies have large spikes in the amount of raw material that they retain in the manufacturing line. This is because both strategies prioritize process 2 over process 1 and this leads to AGVs dumping raw material when they reach process 2 to be able to service process 2. When the amount of raw material in the system reached a threshold which is when the spike is at the highest point, the system stops introducing new material into the system and uses the material stored in the manufacturing line.

In essence, with respect to throughput, the Nearest Robot in Context strategy performs 7% and 1% better than the Baseline strategy in the 3 and 5 robot case respectively. The Late Commitment strategy exhibits much stronger performance, achieving a 98% improvement in throughput over the Baseline strategy in both the 3 and 5 robot scenarios.



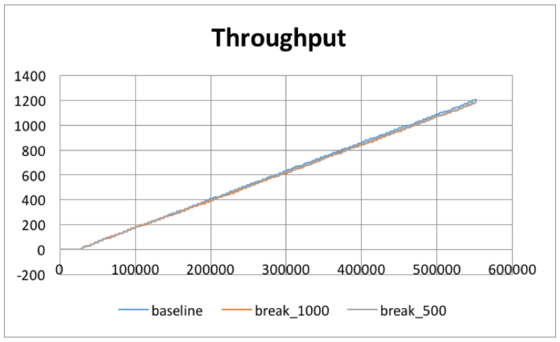
(a) Throughput



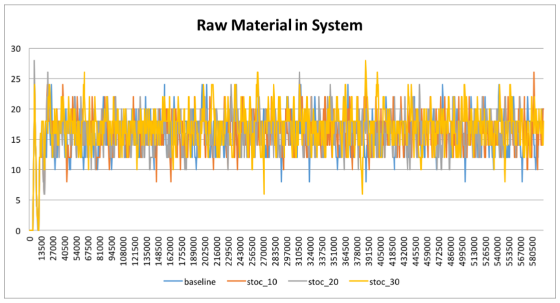
(b) Raw Material in System

Figure 8.3: Performance of system with stochasticity in process times

We have shown that that the Late commitment planner yields significant improvement over the baseline strategy. However, the tests conducted so far have been on deterministic systems.



(a) Throughput



(b) Raw Material in System

Figure 8.4: Performance of system with failed parts

Given that in the real world, machines are stochastic, to obtain a more realistic performance measure of the planner, it is imperative to test it on stochastic systems. To do so, two types of test were performed:

1. **Stochasticity in process times**: The process times for machining parts in CNC 1 and CNC 2 were picked from a normal distribution. The standard deviation was varied to see how the performance varied with larger variations in processing times.
2. **Failure in parts processed**: Failures were introduced in the processing of parts. This was done by increasing the process time of a part to ten times the original processing times.

In the case where stochasticity was introduced in the processing time of the tasks, three scenarios were tested against the performance of the planner in the deterministic setting by changing the standard deviation to 10%, 20% and 30% of the value of the deterministic processing time. The graph in Figure 8.3a represents the throughput of the three scenarios along with that of the planner in the deterministic setting (baseline). It can be seen that the throughput of the planner with the stochastic processing times is nearly identical to that of the planner in the deterministic setting. The graph in figure 8.3b shows the amount of raw material in the system and it can be seen that like in the deterministic setting, the amount of raw material in the system remains stable.

In the case where failure of parts in the machines were simulated, two cases were considered (1) 1 part failing in every 1000 parts machined and (2) 1 part failing in every 500 parts. It can be seen in figure 8.4a that similar to the case with stochastic processing times, the throughput is nearly identical to that of the planner in the deterministic case. The same is true for the handling of the raw material in the system and is shown in figure 8.4b.

Chapter 9

Future work

There are many possible directions for future work:

The current system has only been tested in simulation. An important step could be to integrate the system with a physical manufacturing setup and test the (1) scalability of the ROS communication protocols and (2) the connection of the interface layer to the physical factory floor.

Also, while the current system shows a significant improvement over the baseline implementation, we believe that the system can be improved further by taking advantage of the data that is collected in the data aggregation layer. Using statistical methods to predict failures and improve the models of the machines in the environment will allow for the development of a more opportunistic scheduler that will take advantage of the conditions in the system and proactively schedule the AGVs to keep the factory functioning at near optimal levels.

Further, algorithms may be developed for the processing of data coming in from the factory floor to provide better understanding of the functioning of the factory.

Chapter 10

Conclusions

Industry is rapidly moving towards Industry 4.0 and the advantages that opportunistic scheduling offer.

In this work, we have presented an architecture for integrating cyber physical systems into manufacturing. We have then shown how that architecture can be used in a sample manufacturing problem with an almost 98% improvement over the current process in a baseline Foxconn implementation. Also, due to the reactive nature of the system, we have demonstrated that it is very robust against uncertainty.

Bibliography

[1] E Ackerman. Fetch robotics introduces fetch and freight: your warehouse is now automated. *IEEE Spectrum*, 2015. 2

[2] Manfred Broy. *Cyber-physical systems: Innovation durch softwareintensive eingebettete Systeme*. Springer-Verlag, 2011. 1

[3] Raffaello D'Andrea. Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering*, 4(9):638–639, 2012. 2

[4] Dennis Kolberg and Detlef Zühlke. Lean automation enabled by industry 4.0 technologies. *IFAC-PapersOnLine*, 48(3):1870–1875, 2015. 2

[5] Edward A Lee. Cyber physical systems: Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369. IEEE, 2008. 1

[6] Jay Lee, Behrad Bagheri, and Hung-An Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3:18–23, 2015. 2

[7] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, number 3.2, page 5. Kobe, Japan, 2009. (document)