Resource Reconfiguration Decisions for Distributed Shop Floor Control

Patrick McDonnell
Sanjay Joshi
Industrial and Manufacturing Engineering
Pennsylvania State University
207 Hammond Building
University Park Pa., 16802

Abstract

Past applications of artificial intelligence techniques to shop floor control tasks are reviewed. In the area of resource allocation in distributed manufacturing systems, the issue of resource reconfiguration is identified as a neglected problem. A game theoretic planning model suitable for use in a distributed shop floor control environment is introduced. Autonomous machine controllers consult the planning model to evaluate reconfiguration decisions as the controlled machine resource becomes idle. The results of the evaluation, an equilibrium play of the setup game describing the reconfiguration decision, dictate any resource reconfigurations undertaken by the machine controller.

Introduction

While the emergence of Agile and rapid response manufacturing environments demands flexible and adaptable production facilities. the cost and complexity associated with developing shop floor control software using conventional techniques has hindered the capabilities and implementation of integrated shop floor control systems. As information rich environments in which individual problems and entities are confounded with one another, where large problem size prevents the application of global optimization procedures, and where unexpected events and system disturbances are common place, manufacturing shop floor environments mirror the challenges found in other large scale systems. In other domains. particularly in artificial intelligence, the difficulty of managing large scale engineering systems has encouraged the development of decentralized and distributed procedures for system management. While similar trends are evidenced in recent manufacturing systems research, a comprehensive set of tools does not yet exist for distributed control of a shop floor environments.

This paper looks at how developed and emerging artificial intelligence techniques map into the domain of distributed shop floor control, identifying the parallels and differences between the original AI and current manufacturing applications. We find that for low level tasks AI planning techniques provide good coverage of the problems to be solved. but that the manner in which to apply specific techniques becomes less clear as the scope of the considered problem stretches to include issues of system wide performance. Often this problem derives from characteristics of the shop floor control domain which are not encountered in the original application. However, in some cases important long range issues have not been well addressed for distributed environments. We identify one such issue. production resource reconfiguration, which is essential to the operation of manufacturing facilities and propose a game theory based planning framework suitable for implementation in a distributed shop floor control environment.

Shop Floor Control

Shop floor control systems must address diverse problems ranging from the allocation of production resources and the error free coordination of physical actions to the scheduling of maintenance and the replenishment of consumable resources. These problems must be solved simultaneously for multiple parts and multiple resources in the face of uncertainty caused by machine failures, transport delays, tool breakage and other disturbances. Given these difficulties it isn't surprising that empirical surveys(Ranta & Tchijov 1990) find most flexible manufacturing systems to be small in size, unsophisticated, and ironically inflexible in the face of change.

Traditional Shop Floor Control

Traditional approaches decompose the shop floor control problem by implementing hierarchical control architectures(Dilts, Boyd, & Whorms 1991), propagating control decisions down a hierarchy of tightly bound controllers and percolating shop floor status information

up the hierarchy. At each level in the hierarchy controller responsibilities fall into three categories: planning, scheduling, and execution (Joshi, Wysk, & Jones 1990). Informally, planning activities determine what tasks should be performed, scheduling orders the tasks to be performed, and execution implements the identified tasks. With the master-slave relationships characteristic of hierarchical architectures the planning and scheduling capabilities of controllers close to the shop floor are often emasculated, replaced by direction from superior controllers in the hierarchy.

Events	Time Scale	Technologies
Master Plan	Days	MRP
Failure	Hours	Gantt Chart
Configuration		Stochastic Models
Replenishment	Minutes	Dispathcing Rules
Routing		Auctions
Coordination		Automata
Operation	Seconds	Petri Nets
Sub Task		NC Program
Process Sensing	SubSecond	PID Control
Motor Control		

Figure 1
Shop Floor Events

As noted by Kimemia and Gershwin (Kimemia & Gershwin 1983) and illustrated in Figure 1, the events and tasks which must be accounted for in planning occur at varying frequencies. The highest frequency events are those associated with physical operations on a part, followed by events associated with resource allocation, changes in setups and reconfiguration of production resources [1], machine failures, and finally the production planning horizon of the facility. High frequency events are subject to local disturbances (i.e. tool breakage, execution delays) and are often confined to individual entities while disturbances to low frequency events are often a convolution of system wide effects. The demarcation between levels of control in hierarchical environments mirrors the divisions in the frequency domain of shop floor events: the lowest levels in the control hierarchy addressed the highest frequency events while higher levels in the hierarchy are able to address low frequency issues.

Distributed Shop Floor Control

Problems with the rapid increase in system cost and software complexity with the number of machines in the system (Ayres 1992), concerns about fault tolerance of hierarchical systems (Duffie, Chitturi & Mou 1988), and the difficulties of modifying systems once built (Crowe 1992)(Duffie 1990) have prompted the development of distributed heterarchical approaches for shop floor control. Duffie, an early proponent of heterarchical manufacturing architectures, suggested several principles for decomposing a shop floor control system into autonomous entities including the elimination of master/slave relationships between entities, local retention of locally generated information, and the establishment of delayed as opposed to permanent relationships between entities. While control entities should be predisposed to cooperation with other control entities in the system, they should not assume that any other entities will be available or willing to cooperate. Time-critical decisions and responses should be in the domain of individual entities and entities shouldn't rely upon global knowledge to formulate actions.

Decomposition of a control system according to Duffie's principles typically finds individual control entities associated with individual parts, with individual processing resources, and with support information processes(Duffie 1990). Similar approaches to controller decomposition can be found in so called 'egalitarian' control structures(Tchako et al. 1994), 'responsibility-based' manufacturing (Adamides 1995), and the simulation studies of Shaw (Shaw 1987), Maley (Maley), Upton(Upton 1992), and Lin(Lin & Solberg, 1992). Under such an approach control entities are responsible for individually arranging solutions to local problems both through execution of actions within the local domain of control and also through short term arrangements with other control entities with necessary skills and/or resources. Auction algorithms akin to the Contract Net proposed by Davis and Smith(Davis & Smith 1983) are often used to arrange temporary cooperation between entities in the system.

[1] In Gershwin's original decomposition, changes in setup occurred at the highest frequency and were considered negligible events. However in systems without perfectly flexible machinery this isn't the case.

failures and are easily extensible. As individual control entities are modular and smaller in scope, software complexity and cost is reduced as well. However, it should be noted that a system of heterarchical controllers must address the same issues addressed in hierarchical systems. including the planning, scheduling, and execution of low frequency events such as set up changes and responses to machine failures. Without centralized entities to analyze and enforce globally optimized action, these events may be difficult to address. To date these issues have not been adequately addressed in the context of distributed manufacturing systems with autonomous controllers.

AI and the Shop Floor Control

Concepts and techniques derived from work in artificial intelligence have been applied to a wide range of applications related to shop floor control. The following discussion provides a sample of AI techniques applied to shop floor control activities. For well defined tasks, these techniques provide a complete coverage of the problem. However as the scope of the problem increases to include issues of system level resource management, the mapping of AI techniques onto shop floor control isn't completely specified.

Application of AI Techniques

Traditional nonlinear planning tools have successfully been applied to problems such as low level operation sequencing (Shaw 1987) and assembly planning(Kokkinaki & Valavanis, 1995). In these applications, sequences of atomic activities are constructed to take a problem from a well defined initial state to a well defined goal state. Generally the control entity constructing the plan has complete control over the specification or implementation of the plan and disturbances to the task occur within the domain of influence of the controller.

Contract Net type auction protocols (Davis & Smith 1983) have found wide use as an effective mechanism for distributing production tasks to production resources in distributed and heterarchical environments (Parunak 1988) (Shaw 1987) (Upton 1992). These applications have considered an abstraction of the resource allocation problem in flexible manufacturing, focusing on the allocation of primary production resources (machines). At this level of abstraction, the problem of resource allocation is strongly analogous to the problem of allocating jobs in distributed computing environments(Ferguson, Yemeni & Nikolau 1988)(Litzkow, Livny & Mutka 1988) and system characteristics mirror so called computational ecologies (Waldspurger, et al. 1992)(Hogg & Huberman 1991).

Simulation studies (Upton 1992) of auctioning algorithms indicate robust performance with auction based resource allocation, but it is still unclear how to properly specify the goals of individual entities and how best to assign and supply the secondary resources (i.e. tooling, fixturing) required for production. As detailed in the following section, closer inspection of the problem reveals important differences between the problem of resource allocation in the original distributed computing domain and the proposed shop floor domain.

The advent of agent based systems provides another example in which AI technologies provide inspiration for research in shop floor control. However, at the present stage of development it is unclear how to best apply agent technologies to the shop floor. While ostensibly attuned to agent based principles, the implementations referenced in the second section are different than systems inspired by DAI multi-agent architectures (Jennings 1995)(Berry & Kumara, 1995). In the first instance, controllers with planning, scheduling, and execution abilities act as 'agents' while in the case of 'multi-agent' architectures agents have capabilities focused upon solving a specific class of problem. For instance, individual agents may be responsible for path planning, plan implementation, information retrieval, learning, or monitoring of the environment. It is conceivable that the same agent may be used to plan paths for all of the parts in the system. In any case, many questions remain regarding how to decompose planning and scheduling analysis to take advantage of the distributed nature of agent based systems and to avoid the problems of complexity and tractability which undermine conventional approaches to shop floor control.

Contrast of Resource Allocation Problems

The case of resource allocation illustrates the differences which often arise between original domains of AI applications and the domain of shop floor control. At an abstract level the analogy between CPU allocation in distributed computing environments and machine allocation on the shop floor is valid, but at a more detailed level the analogy is incomplete. As depicted in Figure 2, computing and manufacturing environments differ in the capabilities of system resources, the prerequisites to processing, ease of job transportation, and the frequency and duration associated with processing events.

In computing environments capabilities of processors are generic. Though processing power and speed may vary.

processors are generally capable of performing all foreseeable tasks and there is little penalty for switching between classes of jobs. In contrast, many manufacturing environments are characterized by processors with specialized capabilities. While flexible machine tools exist, there is a cost associated with changing from one class of job to another.

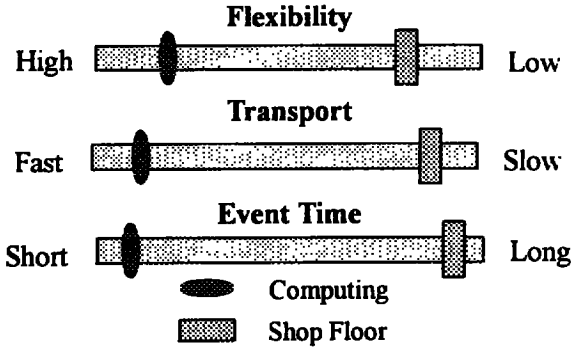

Figure 2
Shop Floor and Computing Contrasts

In networked computing environments transportation of jobs from one site to another is easy. While congestion may delay the transmission of a job from one site to another, failures of the transmission may be readily detected and may be corrected by reinitiating the transmission of the job. In manufacturing environments, transport of resources may take significant amounts of time and disturbance to the transportation task may not be detectable at either the origin or the destination of the transport task.

In computing environments, decision making is under strong timing constraints. Extensive optimization of routing decisions may not be possible due to the volume of traffic which must be considered and the short processing times associated with many jobs. Though shop floor control decisions are subject to more complex constraints, more time may be devoted to individual allocation decisions due to the longer processing times and lower frequency of events at individual machines.

In short the problem of resource allocation in manufacturing environments is more complex than the analogous problem considered in distributed computing environments. However, the timing constraints placed on 'real time' resource allocation decisions are not as strict as those encountered in distributed computing environments.

Reconfiguration Planning

As noted in the previous section, resource reconfiguration is an important issue in the management of manufacturing systems which has not been well addressed for distributed environments. We consider a heterarchical system in which individual machines must make reconfiguration decisions independently, where machine entities can't explicitly coordinate their actions, and where the benefit of a given configuration is a function of the demand as well as the configuration of the other machines in the system.

When resources have flexible capabilities which can be reconfigured at some cost, entities controlling resources are confronted with a strategic decision regarding which configuration to select. Switching between configurations is an expensive, time consuming task and the utility of a specific configuration is a function of the demand for the configuration as well as the nature of the configurations selected by other resources in the system. Under our approach, a machine entity uses a game theoretic model of the system to evaluate whether to maintain the existing configuration or to change to a new configuration. Given the desire to maintain the independence of machine controllers, the set up decision is modeled as a noncooperative game. When confronted with an opportunity to change configurations, the machine controller constructs and evaluates a game based upon its perception of the current system configuration. The action to be taken is defined by the selected equilibrium solution of the game.

Application of Game Theory

Game theory is a collective term describing a set of analytical tools used to analyze the behavior of decision making entities in environments where the objectives and decisions of individual decision makers interact. In the parlance of game theory, a 'game' represents a decision making situation and is characterized by a set of decision makers (players) each of whom have a set of feasible decisions (strategies) and individual preferences over the outcome of the game. In general, game theory is used to model situations in which individuals don't have complete control over the outcome of the game yet possess enough influence to alter results through individual action.

The goal of game theoretic analysis is to describe how rational decision makers should act in the decision making situation modeled by the game. 'Solutions' to a game take the form of equilibrium plays, combinations of feasible decisions such that no decision maker has the

individual incentive to change his/her decision given the decisions of the other players. The concept of equilibrium play maps well into domains where autonomous agents make decisions according to local goals. In formulating a proposed action, game theoretic analysis distinctly accounts for the likely actions and reactions of other agents in the system without requiring explicit interaction between the agents involved in the game. Furthermore, engineering systems provide a good environment for the application of game theory. While one must hypothesize about the rationality of human decision makers or the manner in which an agent selects between multiple equilibria in human systems, the designer of an engineering system can embed rational decision making into agents, can dictate policies for selecting equilibria and can manage the distribution and use of knowledge. As shop floor control systems are closed, one need not worry about the 'morals' and intentions of entities in the system (Rosenschein 1994).

Game Definition and Solution

A game may be described in either Normal or Extensive form. Normal form representations model a situation as a single set of simultaneous decisions while Extensive Form representations include a topology describing the dynamics of a sequential decision making process. In the most basic Normal Form situation, where players make a single set of simultaneous decisions, a game G is defined as an 5 tuple:

(1) $g = G(P,A,C,O,U)$

where P is a set of players $p_1,...p_N$

A is a set of feasible strategies available to players

C is a set of Consequences or outcomes of the game

O is a function, $O:A_1xA_2x..A_N\text{-->}C$, which maps player's actions onto outcomes of the game

U is a set of preference functions, $U_1...U_N$, which order the outcomes of the game for each player.

Each player $p_i \in P$ has a an associated set of strategies or actions, $A_i \subseteq A$, and a_i represents the strategy selected from A_i by player p_i.

Game theory predicts that rational players should select a combination of strategies such that no player has an individual incentive to deviate. The strategy selected by a player is a best response to the strategies selected by other players in the game. Though several variations and refinements of equilibrium conditions have been developed, the Nash equilibrium is the key solution concept. Mathematically, a set of strategies $(a_1^*,a_2^*,...a_N^*)$ which constitute a Nash equilibrium satisfy the following set of inequalities:

(2) $U_i(O(a_1^*,...,a_i^*,a_N^*)) \geq U_i(O(a_1^*,...,a_i,a_N^*))$

for $i=1..N$ and $a_j \in A_j$

The a_i in the above expression may be replaced by α_i where α_i represents a randomization over a feasible subset of a_i in A_i and is referred to as a mixed strategy.

Set Up Game Description

The game situation to be considered in this paper models an environment in which several uncorrelated job streams are serviced by several identical, reconfigurable machines. Based upon a local perception of the environment, the machines must autonomously determine when to initiate a change of setup. As illustrated in Figure 3, the machines are managed independently and may only service a single job stream at a time. In cases where the number of job streams outnumbers the number of servers and cases in which system parameters fluctuate, steady state analysis is inappropriate for system control

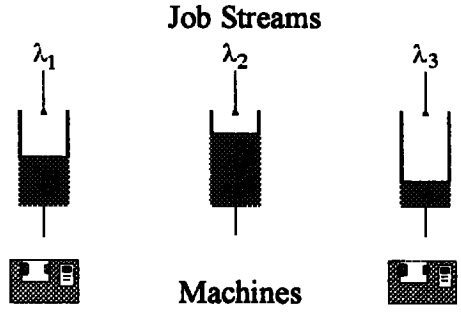

Figure 3
Setup Game Scenario

Machines are faced with a reconfiguration decision every time the machine becomes idle. Upon completion of processing, a machine controller instantiates a Normal form game to describe the current status of the shop floor. Players in the game represent machines in the system, and the strategies open to players are the set of feasible processing configurations. As maintaining the existing setup is a feasible strategy, the set of strategies available to players is never empty.

Payoffs in the game are derived from the existing backlog of and projected demand for servicing the job streams in the system. from the existing configuration of machines in the system, and from the costs associated with changing setups. The payoffs represent the discounted long term costs and benefits of the setup decisions. The quality of decisions is impacted by the metric chosen for measuring cost as well as the accuracy with which the long term costs of a decision are measured. However, as discussed in greater detail below, it is difficult to accurately and efficiently derive these long term payoffs.

A Normal form representation doesn't fully capture the dynamics of the shop floor situation: in reality a machine will not be reconfiguring to the selected configuration for the remainder of service. However, a complete extensive form representation is far too complex to be dynamically created and evaluated. Furthermore, in the event of a poor decision, the machine may recover at the next evaluation cycle by evaluating a new game model based upon the new state of the system. In effect the cycle of process. evaluation, and setup implements a form of feedback control on the system.

Execution of the Setup Game Model

As illustrated in Figure 4, execution of a reconfiguration decision using the setup game model occurs in four stages: model initialization, model evaluation, equilibrium selection, and decision implementation. The first stage involves the creation of a setup game to represent the current state of the system. Payoffs may be derived from analytical expressions, heuristics or, as described in the following subsection, from a long term planning model. The second stage of the procedure involves solving the setup game to find a set of feasible equilibria. In the present stage of development, only systems with two parallel machines are considered for evaluation. As a result, the Lemke-Howson algorithm (Lemke 1965) is used to generate a complete set of feasible equilibria as candidate solutions for the game. In the third stage of operation, players must select a single equilibrium for implementation. For the approach to be successful both machines must consistently select a single equilibrium of the game for implementation. The procedure is completed by physically implementing the course of action suggested by the game decision.

System Control

Figure 5 illustrates the control of a system using a simple version of the set up game model. The system consists of three independent job streams and two individual machines. In this case, machines evaluate a reconfiguration upon coming idle. But approximate the payoffs in the game by assuming a constant arrival rate of work to each of the job streams and accounting for only a single reconfiguration decision by each machine over the planning horizon. In the case illustrated in Figure 5. interarrival times and service times of jobs follow exponential distributions. The average arrival rates for streams 1, 2, and

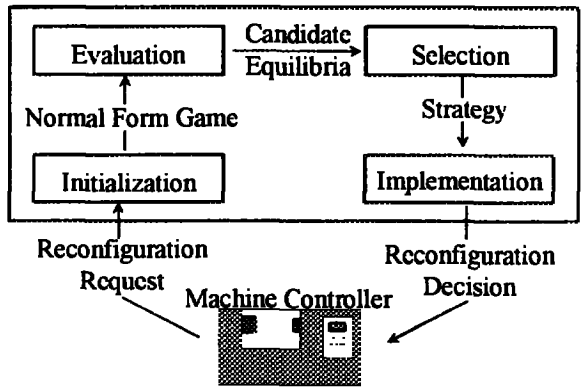

Figure 4
Setup Game Implementation

three are 0.1 jobs/min.. 0.08 jobs/min. and 0.05/min. respectively. Average service times for jobs in streams 1. 2. and 3 are 6.67 min., 8.33 min. and 14.3 min. respectively. 6 minutes are required to change to setup 1. 10 minutes are required to change to setup 2. and 5 minutes are required to change to setup 3. As can be seen from the figure. service continues on the tended job streams until the backlog of work in the untended job stream begins to cross a threshold. at which time a reconfiguration of one of the machines is initiated.

Long Term Planning Model Approach

While the heuristic approach described in the previous subsection is desirable from the standpoint that games may be rapidly initialized, it may be subject to poor decisions due to inaccuracies in the model. Inaccuracies arise from the difficulty of specifying long term costs of actions and the myopic nature of a single stage normal form representation which ignores the dynamic nature of decision making. To address this deficiency. we are investigating the development of a long term planning model to support the specification of setup games.

The planning model is designed as a background process to run concurrent with the machine controller. As the planning model operates, it continuously refines estimates

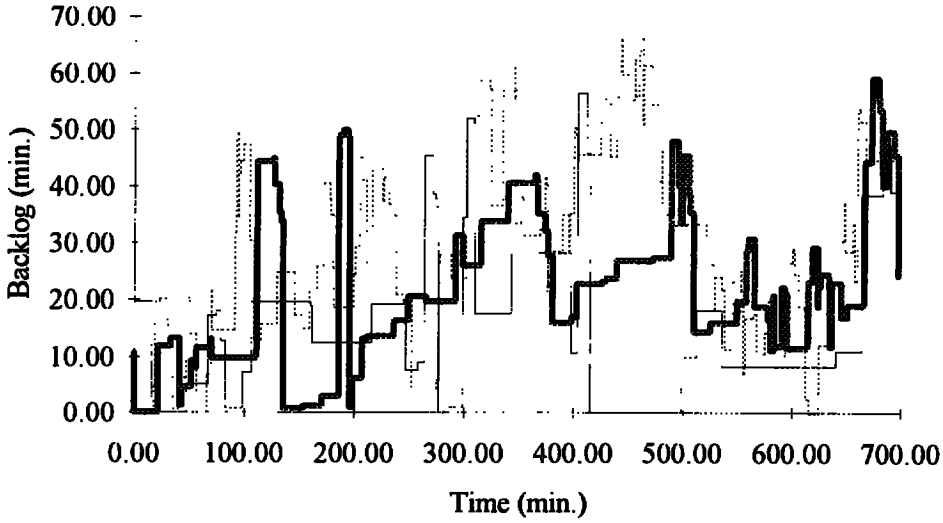

Figure 5
System Control Under the Setup Game Model

of the discounted long term expected cost of entering different states of the system. When confronted with a reconfiguration decision, the machine controller instantiates a normal form set up game based upon the current cost estimates of the long term planning model and follows a procedure analogous to that described in section on the execution of the setup game model.

As the costs from the long term planning model reflect the long term expected cost of entering a state, including the contributions of future transitions out of the state, the estimates account for both the immediate and the future decisions of machines. This approach is similar to the use of sub game perfection in solving extensive form games. In this case, the costs of early decisions in an extensive form game tree are calculated by backwards substitution of the equilibrium solutions of 'child' subgames for the payoffs of alternatives in the 'parent' game.

The planning model estimates the stationary expected cost of entering any of the discrete states of the system. For the purposes of the planning model, the system state space is defined by the Cartesian product of backlogged job stream demand and the feasible configurations of machines in the system. Transitions between states in the planning model are defined by the arrival rate of new jobs, the service rates of machines in the system, the reconfiguration rates of machines in the system, and the reconfiguration decisions of the machine controllers for each state in the planning model.

Assuming that the true cost of transitioning to neighboring states are known, the expected reconfiguration decisions for machine controllers in each state of the model can be calculated by evaluating a game associated with the state. The selected equilibrium solution of the game describes the probability that each machine will reconfigure to a new set up conditioned upon the ability of a decision making opportunity. However, the true costs of entering neighboring states are unknown and must themselves be estimated.

In an approach similar to value iteration algorithms and the Real Time Dynamic Programming method described by Barto et al.(Barto, Bradtke, & Singh 1995), the long term planning model improves estimates of state costs by iteratively evaluating and updating the cost estimates of states in the system model. At the beginning of each planning iteration, where a planning iteration is defined as the period between reconfiguration decisions, the long term planning model begins with an estimate of the stationary expected cost of being in a given state. The planning model iteratively evaluates each state in the model, updating the cost estimate for the state by reformulating and reevaluating the setup game associated with the state using the current discounted cost estimates of neighboring states. As the reevaluation process continues the quality of the cost estimates improve as does the quality of the reconfiguration decisions themselves.

Conclusions

We have reviewed the application of several AI techniques to problems of shop floor control and found that at higher levels the problem of control is not comprehensively addressed. In particular we identify resource reconfiguration as a problem which has not been adequately addressed for distributed shop floor control environments. A game theoretic approach to resolving reconfiguration decisions is introduced as a suitable approach for distributed and agent based environments. Payoffs for setup games may either be derived from approximations of long term decision costs or from a long term planning model which iteratively develops improved estimates of long term expected cost at different states in the system.

Though the long term planning model is computationally intensive, the low frequency of reconfiguration decisions at individual machines suggests that the long term planning model will have adequate time to improve cost estimates between reconfiguration decisions. Furthermore in systems in which the system state exhibits slow drift, each planning iteration can build upon and improve the cost estimates generated by previous planning iterations. Finally, the computational burden may be lessened by concentrating evaluations on states within a given neighborhood of the current system state.

Acknowledgments

This research was supported under a Deprtment of Energy Integrated Manufacturing Fellowship awarded to Mr. McDonnell and under NSF award DDM-9158042 awarded to Dr. Joshi.

References

Adamides, E., "Coordination of Distributed Production Resources for Responsibility Based Manufacturing". *Journal of Intelligent Manufacturing*, Vol. 6, pp.415-427 (1995).

Ayres, R., "CIM a Challenge to Technology Management", *International Journal of Technology Management*, Vol. 7: Nos. 1/2/3 , pp. 17-39 (1992).

Barto, A., Bradtke, S. and Singh, S., "Learning to Act Using Real-time Dynamic Programming". *Artificial Intelligence*, Vol. 72, pp.81-138 (1995).

Berry, N. and Kumara, S., "A Multi-Agent Architecture for Automated Manufacturing System Control". *Proceedings of the Workshop on Improving Manufacturing Performance in a Distributed Enterprise*, pp. 99-108 (1995).

Crowe, T., "Integration is Not Synonymous With Flexibility", *International Journal of Operations and Production Management*, Vol. 12, No. 10, pp. 26-33 (1992).

Davis, R. and Smith, R., "Negotiation as a Metaphor for Distributed Problem Solving", *Artificial Intelligence*, Vol. 20, pp. 63-109 (1983).

Dilts, S, Boyd, N., and Whorms, H., "The Evolution of Control Architectures for Automated Manufacturing Systems". *Journal of Manufacturing Systems*, Vol. 10, No.1 (1991), 79-93.

Duffie, N., "Synthesis of Heterarchical Manufacturing Systems", *Computers in Industry*, Vol.14, pp.167-174 (1990).

Duffie, N, Chitturi, R., and Mou, J., "Fault Tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities", *Journal of Manufacturing Systems*, Vol. 7, No.4, pp. 315-328 (1988).

Duffie, N. and Piper, R., "Non Hierarchic Control of a Flexible Manufacturing Cell". *Robotics and Computer Integrated Manufacturing*, Vol. 4, No. 1, pp.175-179 (1987).

Ferguson, P, Yemeni, Y., and Nikolaou, "Microeconomic Algorithms for Load Balancing in Distributed Systems", *IEEE International Conference on Distributed Computer Systems*, pp. 491-499 (1988).

Hogg, T. and Huberman, B., "Controlling Chaos in Distributed Systems", *IEEE Transactions on Systems Man and Cybernetics*, Vol. 21, No. 6, pp. 1325-1332. (November 1991).

Jennings, N., "Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems Using Joint Intentions". *Artificial Intelligence*, Vol. 75, pp. 195-240 (1995).