Using Decision-Theoretic Planning Agents in Market-Based Systems

Michael Brydon

Faculty of Business Administration and Centre for Systems Science
Simon Fraser University
Burnaby, British Columbia, Canada
mjbrydon@sfu.ca

Abstract

This paper examines a number of theoretical and practical issues concerning the use of decision-theoretic planning to implement agents in market-based systems. The markets considered here result from the decomposition of complex, intra-organizational resource allocation problems such as manufacturing scheduling. Although these problems can be formulated as monolithic optimization problems, they tend to be much too large to solve in practice. Markets provide a means of decomposing large resource allocation problems and distributing the computation of a solution over many processors.

An important precondition of efficient markets is agent-level rationality. Decision theoretic planning can be used to implement economic rationality and thus decision theoretic planning agents fit well into market-based approaches. The primary challenge in building decision theoretic planning agents is the size of the agent's state space. Although market-based decomposition results in agent-level problems that are much smaller than the original resource allocation problem, the price mechanism used to achieve independence of the agent-level problems requires that the agents plan over a large number of different resource contingencies. This requirement exacerbates the state space explosion that characterizes decision-theoretic planning. The application of state space reduction techniques such as structured dynamic programming and reachability analysis are shown to yield significant reductions in the effective size of the agent-level problems and thereby increase the applicability of decision theoretic planning techniques in market-based systems.

1 Introduction

Real-world resource allocation problems such as scheduling production machinery in large manufacturing facilities could benefit greatly from the application of decision-theoretic planning techniques. Such allocation problems are characterized by uncertainty and complex trade-offs between objectives such as time, flexibility, and cost. However, the standard techniques used to allocate resources rely on numerous simplifying assumptions, such as deterministic outcomes and single-attribute objective functions such as minimizing the tardiness or makespan of a set of jobs [23].

A decision-theoretic planning system for manufacturing scheduling would explicitly manage the uncertainty in the production environment and would permit managers to express their objectives in terms of complex, discontinuous, multiattribute utility functions. Unfortunately, the richness of the decision-theoretic planning approach leads to a well-documented explosion in the number of states required to represent the problem [5, 11]. Several techniques, such as state aggregation [5, 6, 12], and reachability analysis [1, 4] have been used successfully to delay the onset of impractically large state spaces. However, it is unlikely that such techniques by themselves are sufficient to permit the formulation and solution of large, industrial-scale decision-theoretic planning problems. Instead, some form of problem decomposition is required [10, 19].

A very general approach to problem decomposition that is attracting a great deal of research interest is based on the notion of agents interacting within markets [8, 9, 21, 33, 34]. The purpose of this paper is to examine the suitability of decision-theoretic planning agents for certain types of market-based systems. Section 2 begins by defining the environmental context of the resource allocation problems considered in this paper. Section 3 works backwards from economic theory and the characteristics of the original resource allocation problem to identify the core requirements for the market-based agents. The suitability of decision-theoretic planning for implementing market-based agents is examined in Section 4. The fundamental question addressed in the section is not whether decision-theoretic planning is an appropriate framework for fulfilling the requirements imposed by economic theory. Instead, the important question is one of computational practicality. Although the agent-level planning problems are exponentially smaller than the undecomposed problem, they typically remain far too large to represent and solve using conventional dynamic programming techniques (see [24]). In order to be solvable in practice, the agent-level planning problems must possess internal structure that can be exploited by state space reduction techniques. The empirical evidence presented in this section suggests that certain important classes of resource allocation problems do possess internal structure that can yield significant reductions in the effective size of the agent-level planning problem. Section 5 concludes with some general observations regard-

ing the use of decision-theoretic planning agents in market-based systems.

2 Market-Based Decomposition

A distinction is often made in the distributed artificial intelligence (DAI) literature between *cooperative distributed problem solving* (CDPS) environments and *multiagent systems* (MAS) environments [3, 15]. The critical difference between CDPS and MAS environments is the existence of a global utility function. In CDPS environments, a global utility function exists; the challenge is to decompose the problem such that the maximization behavior of the agents results in maximization of the global utility function. The final utilities of the agents themselves are irrelevant since the agents exist solely to facilitate the distribution of computation.

In contrast, agents in MAS environments are typically used to model the distributed, multiagent structure of a naturally decentralized problem [28]. For example, agents can be used to represent different firms in a complex supply chain coordination problem. Since the fundamental principles of decision theory prohibit the aggregation or comparison of utilities across individual agents [25], no meaningful measure of global utility exists in such environments. Identifying a "good" of "fair" solution therefore requires a commitment to a particular game-theoretic bargaining solution [26, 28].

The task of allocating scarce production resources in a manufacturing facility is an instance of the CDPS class. Although the term "cooperative" is misleading in this case (since economic agents are strictly self-interested), intra-organizational resource allocation problems have well-defined utility functions. The task facing the designer of the agents is to decompose the global utility function into agent-level utility functions. In the following section, a decomposition is proposed that achieves an additive relationship between global utility and independence between the agent-level problems.

2.1 Decomposing the Scheduling Problem

A natural way to decompose a resource allocation problem in a manufacturing environment is to create computer-based agents for each part and machine in the production system (*part agents* and *machine agents* respectively). All the costs and revenues in the problem domain can then be allocated to individual agents as shown in Figure 1.

The only source of revenue in the system is the payment that occurs when a completed part is shipped to the next stage in the system's supply chain, such as a final customer, a distributor, or finished goods inventory. Incoming revenue is modeled using a *terminal reward* that is received by a part agent when the part it represents leaves the production system. The size and functional form of the reward are determined exogenously but are known to the part agent.

On the cost side, both part agents and machine agents incur costs. In the case of part agents, each part is allocated the costs of its constituent raw materials as well as the cost

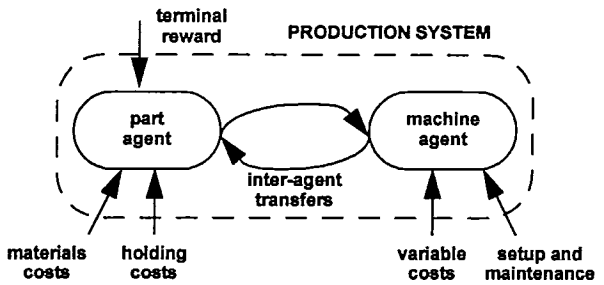

Figure 1: An allocation of global revenues and costs to a part and machine agent.

of holding the part in work-in-process inventory. The costs of wear and tear, consumables, and setups are allocated to individual machines.

The arrows marked "inter-agent transfers" are discussed in more detail in Section 3. At this point, it is sufficient to recognize that all inter-agent transfers are zero-sum and therefore have no net impact on global utility. As a result, the global utility of the production system (revenues - costs) can be expressed as the sum of the individual agent utilities:

$$U_G = \sum_i U_i \qquad (1)$$

To simplify exposition in this paper, we make the assumption that variable, setup, and maintenance costs are zero, This permits us to eliminate machine agents from the decomposed problem and focus on the more complex requirements of part agents.

2.2 Microeconomic Theory and CDPS Environments

The purpose of microeconomic theory is to describe the interaction of individual rational agents pursuing their own selfish objectives [2, 17]. The term rationality, as it is used in the microeconomic context, corresponds to Elster's notion of *thin rationality* [13]. Thin rationality requires only that an agent maximize its utility according to a consistent set of preferences over outcomes and that its reasoning about the expected values of outcomes be consistent with the fundamental axioms of probability theory.

Although the microeconomic formulation of agents is too simplistic to provide an accurate description of human agents, the objective in a CDPS environment is not to simulate reality. Instead, the objective is to distribute computation and aggregate the results so that global utility is maximized. The designer of problem solving systems in a CDPS environment has the luxury of complete control over the implementation and behavior of the agents in the system. This is clearly not the case for the designers of real markets (e.g., the New York Stock Exchange [32]) or problem solving systems in MAS environments. It is therefore possible for a CDPS system to be constructed in strict accordance with microeconomic theory.

The advantage of using microeconomics as a blueprint for a market-based system is that the theory provides some *a priori* insights and guarantees about the overall outcome of the market. Specifically, the First Fundamental Theorem of Welfare Economics states that a competitive equilibrium induces an allocation that is Pareto optimal [17, 16]. That is, if rational, self interested agents are permitted to compete in a market, the outcome at equilibrium is that no agent can be made better off without making some other agent worse off. Although Pareto optimality in the general case is insufficient for global optimality, it is possible to achieve equivalence between the two in cases in which global utility is the sum of agent utilities.

2.3 Maximization Behavior of Agents

According to the decomposition in Figure 1, an agent in a manufacturing system maximizes its utility by attaining its terminal reward while simultaneously minimizing its costs. The cost and rewards that the agent accumulates during its progress through the production system are a function of two things: chance and the agent's control of production resources. It is therefore possible to express agent i's expected utility as a function of its allocation of resources x_i: $U_i = f_i(x_i)$.

Production resources are defined as discrete units of processing time on a particular machine at a particular time. A family of propositional variables Owns(Mj, Tk) can be used to denote whether the agent has exclusive rights to processing on machine Mj at time Tk. The total number of production resources in the system, m, is the number of machines in the production system multiplied by the number of time units in the planning horizon.

The only way that agents can increase their utility is to exchange resources with other agents. Such exchanges can occur between rational agents if and only if neither agent is worse off as a result of the exchange and at least one agent is strictly better off. One means of facilitating such *individually rational* (IR) exchanges between agents is to augment the m-good economy with a *numeraire good.* A numeraire good is a good that possesses the properties of fungibility and divisibility but which has no intrinsic utility to the agents other than its acceptance within the economy as a unit of exchange [17]. Acceptance of the good as a unit of exchange is achieved by adopting and enforcing the convention that all agents express their preferences for resource goods in terms of the numeraire good M rather than their own idiosyncratic and incomparable units of utility. The result is a quasi-linear utility function for agent i:

$$U_i = M_i + \Phi_i(x_i) \qquad (2)$$

To construct (2), the agent scales its utility function to make it linear in good M. The term $\Phi_i(x_i)$ corresponds to the agent's *intrinsic utility* for resource goods. In a manufacturing scheduling environment, the form of $\Phi_i(x_i)$ is determined by the complex interaction of costs and rewards faced by the part. Intuitively, high-value parts with high holding costs and binding deadlines are bound to value certain production resources more than low-value parts that are built to stock.

2.4 Determining Reservation Prices

Determining the rationality of a particular exchange requires agents to know their own private reservation prices for goods. Agent i's reservation price for resource good x^j is the change in the quantity M_i that makes the agent indifferent between an allocation that contains x^j one that is identical in every respect except that it does not contain x^j. Thus given $x_i' = x_i \cup x^j$, $\Delta M_i(x^j) = \Phi_i(x_i') - \Phi_i(x_i)$. Note that the reservation price is symmetrical. It contains no spread between the price at which the agent is willing to buy the resource (bid price) and the price at which the agent is willing to sell the resource if it already owns it (ask price). In addition, the quantity ΔM is independent of the agent's endowment of M. An implication of the quasi-linear utility function is that agents are risk neutral with respect to good M and thus all exchange decisions are made on the basis of marginal changes in M.

An important issue that arises in resource allocation problems is that valuations for resource goods are seldom independent. Two common forms of resource interdependence are substitutability and complementarity. Two goods x^1, x^2 are substitutes if the utility of owning them in the same allocation is subadditive: $U_i(x^1, x^2) < U_i(x^1) + U_i(x^2)$. Conversely, the goods are complements if the utility of owning them in the same allocation is superadditive: $U_i(x^1, x^2) > U_i(x^1) + U_i(x^2)$. Substitutability occurs in manufacturing environments whenever a contract on a particular machine at a particular time can be used in place of a contract on a different machine or at a different time. Complementarity occurs whenever an operation requires more than one unit of processing time or involves precedence constraints.

The deeply embedded interdependencies of goods in resource allocation problems means that the agents invariably participate in a combinatorial auction [18, 27, 29]. In a combinatorial auction, an agent cannot determine its reservation price for a good by considering the good in isolation. Instead, the agent must consider as many as 2^m unique allocations of resources and calculate its reservation price for all transition between all allocations.

For example, in the $m = 2$ economy shown in Figure 2, the agent's reservation price for good x^2 depends on its ownership of x^1. If the agent's initial allocation is $(x^1 = 0, x^2 = 0, M = 100)$, its reservation price for x^2 is \$30 - \$100 = -\$70. In other words, the agent is willing to pay up to \$70 to acquire the resource. However, if the agent's initial state is $(x^1 = 1, x^2 = 0, M = 50)$, its reservation price is only \$0 - \$50 = -\$50. The agent therefore con-

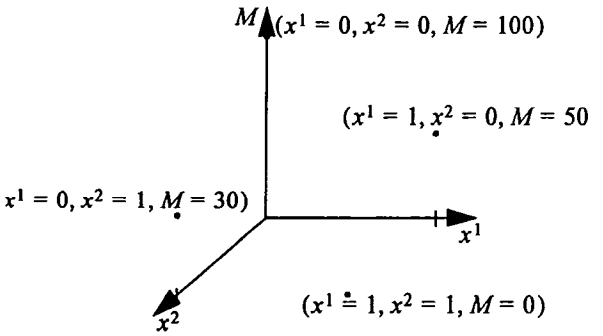

Figure 2: An indifference graph for an economy consisting of two indivisible resource goods (x^1 and x^2) and a numeraire good M.

siders the goods to be partial substitutes since it values x^2 less when it already owns x^1.

3 Interagent Exchange

The decomposition of a resource allocation problem results in a *pure exchange economy* [17]. No production or consumption of the m resource goods or the single numeraire good occurs during the operation of the market. Instead, the agents exchange goods until no further IR transactions are possible. The equilibrium outcome is said to be *efficient* if each good is owned by the agent with the reservation price for the good.

A contract graph [30] provides a convenient means of visualizing the exchanges that can occur in an n-agent m-good economy. Each vertex of the graph describes an allocation of goods to agents. The edges represent exchanges of goods (or contracts) between agents. Figure 3 shows a very simple contract graph for an economy with two agents and two resource goods. The intrinsic utilities of the agents are shown for each vertex as well as the sum of the utilities. Although transfers of M are essential for the execution of the exchanges, interagent transfers of M in a pure exchange economy with quasi-linear utility functions are zero-sum. The total utility of money, $\sum_i U_i(M_i)$, is constant in all allocations and thus the flows of M can be left implicit on the contract graph.

One advantage of the contract graph representation is that it permits the operation of the market to be visualized as local search over allocations. The different auction forms used to implement the market result in different search operators. However, in this case, the agents are risk-neutral

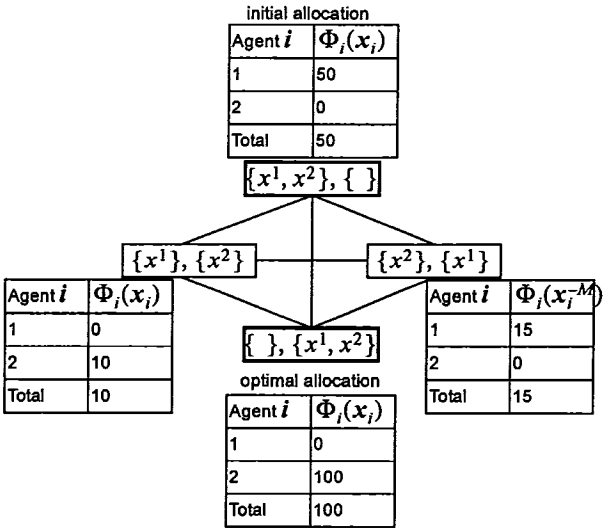

Figure 3: A contract graph showing for a two-agent two-good economy.

with respect to the auction and possess independent private valuations of the goods. Consequently, the major auction forms are equivalent in terms of economic efficiency [20].

Because of the equivalence of auction forms, the market can be implemented as a continuous auction rather than a call auction. In a call auction, a central auctioneer collects reservation prices from potential buyers of a good and awards the good to the highest bidding according to some price function such as first- or second-price [14, 20]. In a continuous auction, agents are always in a position to buy or sell resource goods and transactions are executed without the intervention of an auctioneer.

The primary difference between a call auction and a continuous auction is the number and sequence of contracts required to attain the equilibrium allocation. Since the reservation prices of the bidders are sorted in a call auction, a single contract is all that is required to ensure an efficient outcome. In a continuous auction, the path through the contract graph depends on the order in which the agents interact. Although the distribution of the economic surplus to agents may be different in a continuous auction, the only outcome of interest in a CDPS environment is global utility. A continuous auction therefore provides a convenient means of avoiding the synchronization requirement of a call auction.

4 Decision-Theoretic Agents

Regardless of the auction form used to implement the market, the agents that participate in the market must be able to determine their reservation prices for arbitrary bundles of resource goods. The agents use their reservation prices in two ways. First, an agent acting in the buyer role

needs to identify desirable allocations and determine its willingness to pay for the resources it does not own. Conversely, an agent in the seller role must be able to respond to ask price inquiries from other agents. To complicate matters, agents in a combinatorial auction may be selling and buying within the scope of a single transaction. For example, the contract represented by the horizontal line in Figure 3 has Agent 1 selling x^1 and buying x^2 at the same time.

To this point, nothing has been said about the precise manner in which an agent i determines $\Phi_i(x_i)$ where $x_i \in X_i$ and $|X_i| = 2^m$. However, it is important to recognize that an agent's expected return on its investment in a production resource depends on the way in which the agent uses the resource. Thus, the valuation problem is tightly coupled with an agent-level planning problem.

4.1 Physical and Resource State Spaces

An agent is a computer-based process that represents the interests of an object (in this case, a part) in the real world (in this case, a manufacturing facility). A decision-theoretic planning agent generates a policy π that specifies the agent's best action in every state of the state space of the part it represents [5, 6, 12]. The variables used to represent the "physical" state of a part agent are shown in Figure 4. The actions available to the agent are shown in Figure 5.

Figure 4: State variables for part agents

State variable / family of state variables	Description	Domain
Time	current system time in discrete units	{1,...,*T*}
Status(Op*i*)	the completion status of each operation Op*i* in the part's project plan	{complete, incomplete}
Shipped	whether the part has received its terminal reward	{true, false}
ElapsedTime(Op*i*)	the number of units of processing time already invested in operation Op*i*	{1, ..., max(d_i)}

Figure 5: Actions for part agents

Action /family of actions	Description
Wait	do nothing (the value of Time is incremented)
Process(Op*i*, M*j*, T*k*)	process operation Op*i* on machine M*j* for one unit of time starting at time T*k*
Ship	exit the system and receive a terminal reward

If the part requires three operations and is in a state that satisfies Status(Op1) = complete $\wedge$ Status(Op2) = complete $\wedge$ Status(Op3) = complete $\wedge$ Shipped = false, then the agent's policy states that it should execute the Ship action (and thereby trigger the payment of its terminal reward). Action preconditions ensure that the Ship action is not executed until the part's operations are complete and operations can only be completed by executing Process() actions.

The uncertainties that exist in manufacturing environments regarding the durations of operations, quality problems, and machine breakdowns mean that decision-theoretic planning at the agent-level is seldom trivial. The planning problem is further complicated for market-based agents by contention for scarce resources. An agent cannot simply choose to execute an action such as Process(Op3, M2, T3) since other agents may also want exclusive use of machine M2 at time T3. Meuleau *et al.* [19] address the problem of resource dependency by using heuristic search techniques to merge the policies of the agents into a consistent global allocation of resources.

An alternative approach to centralized dispute resolution is IR exchange based on market prices, as discussed in Section 3. However, in order to determine their reservation prices for goods, the agents must plan over all possible resource contingencies. For every state s_i in the agent's physical state space $s_i \in S_i$, the agent must consider its best action given every possible allocation of resources $x_i \in X_i$. The decision-theoretic planning algorithm generates the agent's optimal policy over the joint physical and resource state space and thereby generates reservation prices as a by-product of its operation.

To illustrate, consider a physical state in which the agent evaluates the expected value of executing the Process(Op3, M2, T3) action. The action has many preconditions including precedence constraints (such as Status(Op2) = complete) and temporal constraints (such as Time = T3). In addition, execution of the action requires that Owns(M2, T3) be true for the agent. However the ownership status of resources is not controllable by the agent through its action set and must be treated as a random variable. During policy generation, the decision-theoretic planning algorithm selects an action for the state $s_i \cap x_i$ where Owns(M2, T3) $\in x_i$ and determines the expected value of that state given the optimal policy π: $V_\pi(s_i \cap x_i)$. The algorithm also selects an action for $s_i \cap x_i'$ where Owns(M2, T3) $\notin x_i'$ and determine $V_\pi(s_i \cap x_i')$. The difference in expected values, $V_\pi(s_i \cap x_i) - V_\pi(s_i \cap x_i')$, takes into account any complex interdependencies between Owns(M2, T3) and other resources and provides the agent's true reservation price for the good while in state s_i.

4.2 State Space Reduction

The obvious problem with planning over resource contingencies is that the effective size of the agent's state space can increase by a factor of 2^m. Even a small problem with

five machines and a planning horizon of ten time units can increase the size of the agent's state space by a factor of 10^{15}. This growth in the state space effectively nullifies the decrease in problem size achieved through decomposition in the first place.

One means of attenuating the explosion in problem size caused by the need to plan over resource contingencies is to use a state space reduction technique such as structured dynamic programming [5, 6, 12]. Structured dynamic programming is based on the observation that not all state information is relevant for all decisions. To illustrate, recall the example used previously in which the Ship action was associated with any state satisfying the clause: Status(Op1) = complete $\wedge$ Status(Op2) = complete $\wedge$ Status(Op3) = complete $\wedge$ Shipped = false. These four state variables are all that is required to choose the optimal action. All other state variables are irrelevant to the decision and can be ignored. Furthermore, if precedence constraints exist between the operations, then only two pieces of information are required to select an action: Status(Op3) = complete $\wedge$ Shipped = false.

The irrelevance of certain state variables in certain decision contexts means that there is no need to enumerate the entire state space to generate the optimal policy. Instead, the relevant state variables can be identified in each decision context and states that are identical with respect to the relevant state variables can be aggregated into *abstract states*. The abstract states are typically represented using a tree structure, as shown by the small fragment in Figure 6. A dynamic programming algorithm can then be applied to the leaf nodes of the tree rather than the individual states. As the policy improves, other state variables typically become relevant and must be included in the tree. For example, the value of the state variable Time is often relevant when the Ship action is executed since the completion time of the part determines whether lateness penalties must be subtracted from the terminal reward. Thus, the number of abstract states grows until the optimal policy is found.

In the best case, the algorithm uncovers numerous sources of irrelevance in the problem structure and generates the optimal policy using an abstract state space that is a small fraction of the size of the fully-enumerated state space. In the worst case, the structured dynamic programming algorithm can yield no reduction in the effective size of the problem but incurs the computational overhead of continually assessing the relevance of state variables. Unfortunately, it is difficult to estimate the impact of the structured approach *a priori*. Thus, the only way to know whether an otherwise unsolvable decision-theoretic planning problem is solvable using structured dynamic programming is to solve it.

Fortunately, problems with the same underlying structure share the same sources of independence and irrelevance. For example, the family of state variables Elapsed-Time(Op*k*) is important for determining the probability that an operation will be complete in the next unit of processing time given the number of units of processing time already invested in the operation. However, these variables always

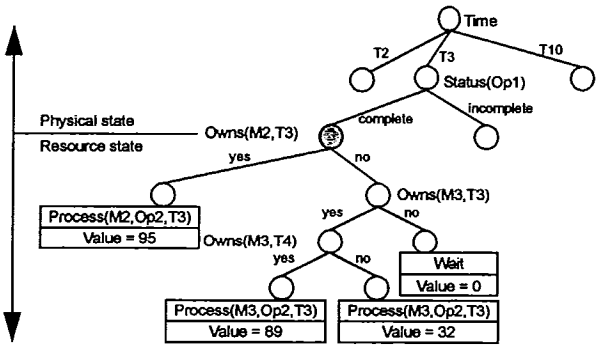

Figure 6: A small fragment of a tree representation of an agent-level planning and valuation problem. The optimal policy and expected value of each abstract state (leaf node) is shown.

cease to be relevant once operation Op*k* is complete regardless of the nature of the operation, the distribution of completion times, or any other problem-specific factor. In such cases, the magnitude of the reductions achieved using structured dynamic programming for one problem instance are typically generalizable to all instances with the same underlying structure.

4.3 Empirical Results

The primary research question addressed in this paper is whether structured dynamic programming yields favorable results when applied to decision-theoretic planning problems formulated over a joint physical and resource state space. To answer this question, a number of small but typical agent-level planning and valuation problems were formulated and solved using a variation of Dearden and Boutilier's algorithm [12] that incorporates non-propositional state variables and a limited form of reachability analysis (see [7]).

The test problems belong to the flow shop class of scheduling problems in which each part requires *n* operations on *n* machines. Each operation Op*i* can only be performed on machine M*i* and precedence constraints require that each Op*i* be complete before Op*i*+1 can be started. The duration of each operation is represented by either a single value (deterministic problems) or a probability histogram (stochastic problems). The global objective is to maximize the total profit of the production system. Interestingly, even very small scheduling problems of this form are difficult to solve exactly. For example, the three-machine, three-part flow shop scheduling problem is know to be strongly NP-hard [23].

The results for a number of deterministic and stochastic problems are shown in Figure 7. In each class, the problem size is varied by changing the length of the planning hori-

zons. As the results indicate, the structured dynamic programming approach does indeed delay the onset of impractically large state spaces. A problem instance with three operations and a planning horizon of 11 time units induces an explicit state space of 10^{14} states. However, the structured algorithm requires fewer than 16,000 abstract states to determine the agent's optimal policy and generate sufficient reservation price information to permit the agent to participate in a combinatorial auction.

The results also indicate that there are limits to the structured dynamic programming approach. The number of abstract state spaces grows at a slower rater than the explicit state space; however, the growth is exponential in both cases. Moreover, the growth in the abstract state space is much faster when the underlying problem is stochastic.

4.4 Structure of the Resource State Space

An analysis of the algorithm's performance on the test problems suggests that the decision-theoretic planning problems faced by market-based agents contains three "structural patterns" that can be exploited by state space reduction algorithms.

The first structural pattern is *temporal irrelevance*. The representation of goods used for the resource state space defines each good in terms of a machine and a unit of time. Naturally, as time progresses, units of processing time that occur in the past can have no impact on the future streams of costs and rewards that accrue to the agent. The structured dynamic programming algorithm recognizes the irrelevance of "expired" resources and thus the size of the agent's resource state space decrease as the value of Time in its physical state space increases.

The second structural pattern, *feasibility*, emerges from the interaction of three common properties of scheduling problems: resource complementarities (multiple resources on one or more machines are required to complete the part), holding costs that are a function of the value added to the part, and a binding deadline. If a part agent does absolutely nothing but wait, it receives no terminal reward and, depending on the problem environment, may not incur any costs. Under these circumstances, the lower bound on the agent's expected utility in any state is zero since it can always do nothing. Accordingly, agents typically price all resource goods at zero whenever they are in a physical state in which there is little chance of attaining the terminal reward before their deadline.

The third structural pattern that occurs in the resource state space is *dominance*. An allocation x_i dominates allocation x_i' if $V_\pi(s_i \cap x_i) = V_\pi(s_i \cap x_i')$ and $x_i \subset x_i'$. Intuitively, the algorithm recognizes non-increasing returns from additional resources. Returning to the example in Figure 6, the left-most branch of the tree corresponding to the allocation x_i = {Owns(M2, T3)} has an expected value of 95. If Owns(M2, T3) is false, the agent can use two units of processing on machine M3 as an imperfect substitute and realize an expected value of 89. However, the algorithm never considers the allocation

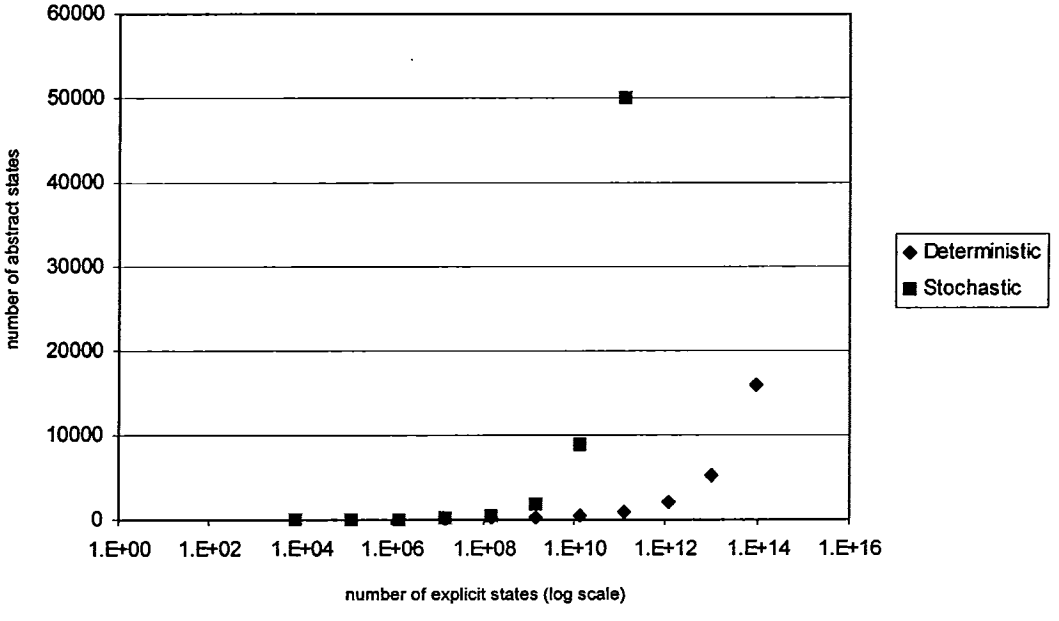

Figure 7: The structured dynamic programming algorithm delays the onset of computational impracticality.

x_i' = {Owns(M2, T3), Owns(M3, T3), Owns(M3, T4)} because all resource goods have a non-zero acquisition cost and the additional goods provide no increase in expected utility for the agent. In short, the agent has nothing to gain by owning x_i' over x_i.

The reduction in the size of the resource state space enabled by these three structural patterns is significant. For example, a problem with three machines and a decision horizon of eight time units generates a resource state space of $2^{3\times8}$ = 16.8 million unique resource allocations. However, the number of allocations actually evaluated in each physical state varies between 508 and 2 according to the frequency distribution in Figure 8. Thus, in the majority of physical states, the agent plans over fewer than 50 distinct resource bundles.

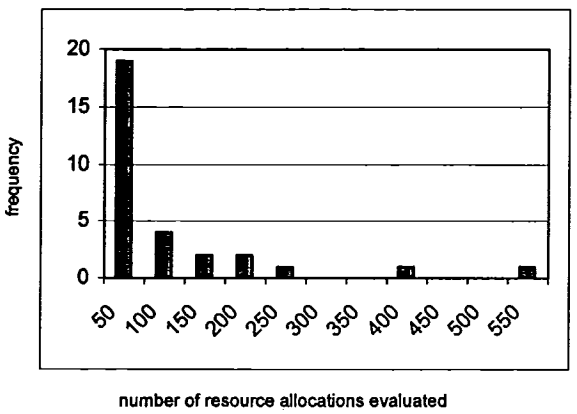

Figure 8: The number of resource allocations considered by the structured dynamic programming algorithm in each physical state depends on the attributes on the state.

5 Conclusions

The use of decision-theoretic planning to generate optimal solutions to large, complex resource allocation problems is infeasible due to the massive states spaces that are required to represent the problems. Although emerging state space reduction techniques such as structured dynamic programming and reachability analysis can greatly reduce the effective size of a decision-theoretic planning problem, these techniques merely attenuate the exponential explosion; they do not eliminate it.

Market-based decomposition of the same large resource allocation problems provides a more realistic opportunity to exploit the power of decision-theoretic planning techniques since the agent-level planning problems are relatively small. Moreover, decision-theoretic planners provide a literal and exact implementation of economic rationality and are therefore ideal for market-based systems based on fundamental microeconomic theory.

The difficulty that arises in practice is due to the price mechanism. Price is a critical element of market-based decomposition since it permits the agents to plan independently and simultaneously in spite of contention for the same scarce resources. However, the price mechanism imposes a significant computational burden on the agents since they must plan over a large number of resource contingencies in order to determine their reservation prices for the resources.

The empirical results presented in this paper suggest that the computational burden imposed by the price mechanism can be reduced by exploiting structural patterns in the resource state space. Specifically, structured dynamic programming can be used to compute the reservation prices of only those resource allocations that are relevant to the agent. This minimal set of reservation prices can be combined to determine the agent's reservation price for any allocation.

An important issue that is not addressed in this paper is the feasibility of determining the equilibrium outcome in the resulting combinatorial auction. The winner determination problem for the auction requires search over a contract graph that is exponential in the number of agents and resource goods [30]. However, a number of heuristic search techniques have been proposed that perform very well, even on very large problem instances [22, 29]. Thus, the main advantage of the market-based approach described here is that it transforms hard, stochastic optimization problems into hard deterministic winner determination problems that are more amenable to heuristic solution techniques.

Given the ultimate requirements for inexact techniques at the market level, one might question the value of using a computationally intensive exact approach such as decision-theoretic planning at the agent level. Although the issue remains unresolved, there are a number of practical reasons why the use of decision-theoretic planning to implement market-based agents is appropriate:

1. **Computation is distributed** — The agents can solve their planning and valuation problems independently and simultaneously. Thus, the computational intensity of the technique is less critical than it would be if the agent-level problems were serialized.
2. **Decision-theoretic planning agents encapsulate uncertainty** — Many real-world resource allocation problems are characterized by risk and uncertainty. Decision-theoretic planning provides an ideal means of incorporating probabilistic information into decision-making. Moreover, the expected values (in the form of reservation prices) generated by the planning algorithm encapsulate the uncertainty in the system. The final winner determination problem can then be formulated as a deterministic problem.
3. **Accurate preference information simplifies the winner determination process** — Search over the contract graph is simplified if each agent has a partially ordered list of resource allocations and intrinsic utilities. This is

especially true if the search is implemented as a continuous auction in which individual agents must initiate exchanges.

4. **The impact of improvements is additive** — Improvements in both hardware and techniques for solving decision-theoretic planning problems have an additive effect in multiagent systems. As such, even incremental improvements can have a significant impact on the feasibility of applying the market-based approach to industrial scale problems.

Of course, this is not to suggest that the decision-theoretic planning techniques used to date in this research are sufficient for industrial-scale problems. Instead of agents with three operations and a time horizon of 11 time units, problems of a useful size require agents to plan over longer time horizons and manage a dozen or so operations. These "useful" agent-level problems are clearly very large. However, they are not so large that they are beyond consideration. Improved techniques for state space reduction, reachability analysis, and approximation could make very large scale market-based systems based on decision-theoretic planning agents a reality.

References

[1] Andrew G. Barto, Steven J. Bradtke and Satinder P. Singh, 1995, "Learning to act using real-time dynamic programming," *Artificial Intelligence*, Vol. 72, No. 1-2, pp. 81-138.

[2] Gary S. Becker, 1976, *The Economic Approach to Human Behavior*, University of Chicago Press, Chicago, IL.

[3] Alan H. Bond and Les Gasser, 1988, *Readings In Distributed Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA.

[4] Craig Boutilier, Ronen I. Brafman and Christopher Geib, 1998, "Structured Reachability Analysis for Markov Decision Processes," in the proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98), Madison, WI (24-26 July) pp. 24-32.

[5] Craig Boutilier, Thomas Dean and Steve Hanks, 1999, "Decision-Theoretic Planning: Structural Assumptions and Computational Leverage," *Journal of Artificial Intelligence Research*, Vol. 11, pp. 1-94.

[6] Craig Boutilier, Richard Dearden and Moisés Goldszmidt, 1995, "Exploiting Structure in Policy Construction," in the proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, pp. 1104-1111.

[7] Michael J. Brydon, 2001, *A Market-Based Approach to Resource Allocation in Manufacturing*, unpublished Ph.D. dissertation, University of British Columbia.

[8] John Q. Cheng and Michael P. Wellman, 1998, "The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes," *Computational Economics*, Vol. 12, pp. 1-24.

[9] Scott H. Clearwater, 1996, *Market-Based Control: A Paradigm for Distributed Resource Allocation*, World Scientific, Singapore.

[10] Thomas Dean and Shieu-Hong Lin, 1995, "Decomposition Techniques for Planning in Stochastic Domains," in the proceeding of 14th International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, PQ, pp. 1121-1128.

[11] Thomas Dean, Leslie Pack Kaelbling, Jan Kirman and Anne Nicholson, 1993, "Planning with Deadlines in Stochastic Domains," in the proceedings of the Eleventh National Conference on Artificial Intelligence, Washington, DC, pp. 574-579.

[12] Richard Dearden and Craig Boutilier, 1997, "Abstraction and Approximate Decision Theoretic Planning," *Artificial Intelligence*, Vol. 89, pp. 219-283.

[13] Jon Elster, 1983, *Sour Grapes: Studies in the Subversion of Reality*, Cambridge University Press, Cambridge, UK.

[14] Richard Engelbrecht-Wiggans, 1983, "An Introduction to the Theory of Bidding for a Single Object," in *Auctions, Bidding, and Contracting: Uses and Theory*, Richard Engelbrecht-Wiggans, Martin Shubik and Robert M. Stark, eds., New York University Press, New York, pp. 53-104.

[15] Nicholas R. Jennings, Katia Sycara and Michael J. Woolridge, 1998, "A Roadmap of Agent Research and Development," *Autonomous Agents and Multi-Agent Systems*, Vol. 1, No. 1, pp. 7-38.

[16] Donald W. Katzner, 1988, *Walrasian Microeconomics: An Introduction to the Economic Theory of Market Behavior*, Addison-Wesley, Reading, MA.

[17] Andreu Mas-Colell, Michael D. Whinston and Jerry R. Green, 1995, *Microeconomic Theory*, Oxford University Press, New York.

[18] R. Preston McAfee and John McMillan, 1996, "Analyzing the Airwaves Auction," *Journal of Economic Perspectives*, Vol. 10, No. 1, pp. 159-176.

[19] Nicolas Meuleau, Milos Hauskrecht, Kee-Eung Kim, Leonid Peshkin, Leslie Pack Kaelbling, Thomas Dean and Craig Boutilier, 1998, "Solving Very Large Weakly Coupled Markov Decision Processes," in the proceedings of the 15th National Conference on Artificial Intelligence, pp. 165-172.

[20] Paul Milgrom, 1989, "Auctions and Bidding: a Primer," *Journal of Economic Perspectives*, Vol. 3, No. 2, pp. 3-22.

[21] Tracy Mullen and Michael P. Wellman, 1995, "Some Issues in the Design of Market-Oriented Agents," in *Intelligent Agents II: IJCAI'95 Workshop on Agent Theories, Architectures, and Languages*, Michael J. Woolridge, Milind Tambe and Jörg P. Müller, eds., Springer-Verlag, Berlin, pp. 283-298.

[22] David C. Parkes, 2000, "*i*Bundle: An Efficient Ascending Price Bundle Auction," in the proceedings of the ACM Conference on Electronic Commerce (EC-99), pp. 148-157.

[23] Michael Pinedo, 1995, *Scheduling: Theory, Algorithms, and Systems*, Prentice-Hall, Englewood Cliffs, NJ.

[24] Martin L. Puterman, 1994, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons, New York.

[25] Howard Raiffa, 1968, *Decision Analysis: Introductory Lectures on Choices and Uncertainty*, Addison-Wesley, Reading, MA.

[26] Eric Rasmusen, 1989, *Games and Information: An Introduction to Game Theory*, Blackwell, London.

[27] Stephen J. Rassenti, Vernon L. Smith and R. L. Bulfin, 1982, "A Combinatorial Auction Mechanism for Airport Time Slot Allocation," *Bell Journal of Economics*, Vol. 13, pp. 402-417.

[28] Jeffrey S. Rosenschein and Gilad Zlotkin, 1994, *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*, MIT Press, Cambridge, MA.

[29] Michael H. Rothkopf, Alexsander Pekec and Ronald M. Harstad, 1998, "Computationally Manageable Combinational Auctions," *Management Science*, Vol. 44, No. 8, pp. 1131-1147.

[30] Tuomas W. Sandholm, 1998, "Contract Types for Satisficing Task Allocation: I Theoretical Results," in the proceedings of the AAAI 1998 Spring Symposium: Satisficing Models, Stanford, CA (March 23-25).

[31] Tuomas W. Sandholm and Subhash Suri, 2000, "Improved Algorithms for Optimal Winner Determination in Combinatorial Auctions and Generalizations," in the proceedings of the National Conference on Artificial Intelligence (AAAI), Austin, TX (July 31-August 2) pp. 90-97.

[32] Robert A. Schwartz, 1988, *Equity Markets: Structure, Trading, and Performance*, Harper & Row, New York.

[33] William E. Walsh, Michael P. Wellman, Peter R. Wurman and Jeffrey K. MacKie-Mason, 1998, "Some Economics of Market-Based Distributed Scheduling," in the proceedings of the 18th International Conference on Distributed Computing Systems, pp. 612-621.

[34] Michael P. Wellman, 1996, "Market-Oriented Programming: Some Early Lessons," in *Market-Based Control: A Paradigm for Distributed Resource Allocation*, S.H. Clearwater, ed., World Scientific, Singapore, pp. 74-95.