JAISCR, 2012, Vol. 2, No. 3, pp. 201 – 214

AN APPROXIMATE DYNAMIC PROGRAMMING APPROACH FOR SEMI-COOPERATIVE MULTI-AGENT RESOURCE MANAGEMENT

Abdeslem Boukhtouta[1], Jean Berger[1], Abraham George[2], Warren B. Powell[2]

[1] *Defence Research and Development Canada-Valcartier*
2459, Pie-XI Blvd North, Quebec, QC, G3J 1X5, Canada
emails: jean.berger@drdc-rddc.gc.ca, abdeslem.boukhtouta@drdc-rddc.gc.ca

[2]*Department of Operations Research and Financial Engineering*
Princeton University, Princeton, NJ 08544
email: powell@princeton.edu

Abstract

Complex problems involving multiple agents exhibit varying degrees of cooperation. The levels of cooperation might reflect both differences in information as well as differences in goals. In this research, we develop a general mathematical model for distributed, semi-cooperative planning and suggest a solution strategy which involves decomposing the system into subproblems, each of which is specified at a certain period in time and controlled by an agent. The agents communicate marginal values of resources to each other, possibly with distortion. We design experiments to demonstrate the benefits of communication between the agents and show that, with communication, the solution quality approaches that of the ideal situation where the entire problem is controlled by a single agent.

1 Introduction

The design and control of complex operations (eg. military operations) often requires the ability to model not only the complex dynamics of physical systems, but also the organization and flow of information and decisions. This dimension captures the fact that complex systems are controlled by multiple agents, each with their own information. The behaviors of the agents will exhibit varying degrees of cooperation that might reflect both differences in information as well as differences in goals. Multi-agent structures may arise because decision makers are distributed spatially (for example, in different terminals in a transportation operation) or functionally (for example, in railroads there is a division of responsibility in the management of locomotives, crews and boxcars). While there is considerable literature on multiagent systems, the vast majority of these studies lack a formal mathematical model and do not consider either the different types of information sharing that arise in this setting, nor the quality of information exchange. Indeed, we address the theme of semicooperative behavior, where different agents work together with different levels of coordination. Cooperation is achieved using information sharing.

The decentralization of decisions among different entities has its origins in the theory of teams ([1], [2]) where members of a team contribute to a common objective by making decisions about different aspects of the system. The individual members need not have complete information about the entire system. A balance is struck between the amount of information available to each member or agent and the associated cost. The assumption in team theory is that the individual members have no conflict of interest. As pointed out in [3], [4] and [5], decentralization of decisions has several advantages such as fault-tolerance and natural exploitation of parallelism, scalability and reliability.

Multi-agent systems deal with the division of tasks between multiple agents with varying levels of communication and cooperation. Reviews and surveys of this field are provided in [6], [7] and [8]. Multi-agent systems are classified into several architectures such as systems with a central control ([9]), autonomous agents that follow identical policies ([10], [11], [12]), multiple agents that interact and learn their own policies ([13], [14], [15]), global world models where agents share information by communicating to a global world model [16] and systems where the agents explicitly communicate with each other [17], [18]. Cao et al. [19] categorizes multi-agent structures based on the method of interaction as those where interaction is brought about via the environment (no explicit communication), sensing (agents sense other agents locally without explicit communication) or communications (directed or broadcast intentional messages with the recipient being known or unknown by the transmitter). The agents may also model other agents using some kind of representation and thereby draw inferences about the actions of other agents.

Most of the techniques that have been developed for solving multi-agent systems estimate the states and learn the actions of the agents using machine-learning algorithms. These estimates are then utilized by each agent in order to make decisions. Previous work on multi-agent systems mention different levels of cooperation between the agents. The system may have multiple agents, but they do not share any information as in [20]. The system could have multiple agents, of which only one would be active in making decisions for all the agents at any single time [9], [21]. Other systems involve non-autonomous agents [22], sometimes with unrestricted information sharing [23]. Implicit sharing of information between agents is also assumed in [24], [25] and [26]. Another approach allows the agents to learn the policies of other "neighboring" agents in addition to their own to bring about a non-direct interaction and optimal behavior of the system as a whole [27], [15] and [13]). Several systems such as in [28], [29], [30], [31], [32] and [8] involve explicit communication between the different agents in the system to bring about coordination.

The presence of a central control is generally assumed in the closely related field of distributed problem solving ([33], [34], [35], [36], [37]) which involves decomposing a large problem into smaller subproblems with the division of information and tasks. These can be either price-directed or resource-directed. Price-directed approaches attempt to determine the right price per unit activity that consumes the resource so that the total amount of resource is distributed optimally. Typical examples include auctioning-type market mechanisms where prices are determined by the decisions of independent agents ([38]) and column generation techniques (although the pricing step requires solving a central pricing problem) as in [39] and [40]. Resource-directed techniques ([41], [42], [43]) allocate resources to activities first and then optimize the activities independently.

This research extends an adaptation [44] and [45] of a multi-agent framework, developed in the context of large-scale freight transportation, for decision-making where agents control resources in a distributed way. The mathematical model uses the dynamic resource transformation problem (DRTP) modeling framework developed in [46] which captures a wide range of realistic, operational details. The algorithmic strategy draws on the methods of approximate dynamic programming [47] which has been found to be quite successful in large, stochastic fleet management problems [48], [46], [49], [50]. The key feature of our model is the absence of a central controller that manages the different agents. Each agent is assumed to only control a subset of resources, and has information about only those resources. The agents optimize their individual objectives autonomously and share the downstream costs of specific decisions to other agents. This shared information enhances the ability of the agents to cover tasks assigned to them, by way of improving the utility of otherwise underused resources.

In this paper, we make the following contributions: 1.) We develop a general, mathematical model of a multi-agent system for distributed, semi-cooperative planning, building on the dynamic resource transformation problem modeling framework. 2.) We categorize the types of information that can be communicated between agents and provide a solution strategy based on approximate dynamic programming for solving the multi-agent model. 3.) We demonstrate experimentally that a semi-cooperative system where the agents communicate information in the form of marginal values of resources outperforms a system with no communication between the agents. The semi-cooperative system demonstrates behavior approaching a system with a single agent control and performs quite well even when the communication between agents is distorted.

This paper is organized as follows. In section 2, we propose a mathematical model for describing a multi-agent system. In section 3, we describe the various categories of information that can be communicated between the agents in a multi-agent system. We discuss the solution strategy and present the algorithm for solving the model in section 4. Then in section 5, we describe the design of experiments to test the benefit of sharing of information among different agents and show that cooperative behavior among agents enable the system to significantly outperform a non-cooperative system and achieve results closer to a system with single-agent control. We summarize our conclusions in section 6.

2 Modeling a multi-agent system

The problem setting involves multiple agents, each controlling a set of resources which have to be utilized to serve demands that arise over a certain period of time. The resources are characterized by attributes. We define a to be a vector of attributes. For instance,

$$\text{a=}\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} \text{Time} \\ \text{Ownership} \\ \text{Location} \\ \text{Type} \end{pmatrix}$$

We further define,

$A =$ Set of all possible resource attributes vectors a.

$R_a =$ The number of resources with attribute a.

We may also model the demands as a second layer of resources with representing the demand attribute space and b denoting the vector of demand attributes. For instance,

$$\text{b =}\begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix} = \begin{pmatrix} \text{Time} \\ \text{Ownership} \\ \text{Location} \\ \text{Type} \\ \text{Expiration} \end{pmatrix}$$

Even though we make no further mention of demands in the context of attributes or attribute spaces, all the elements of the modeling process that we discuss in regard to resources are applicable to demands as well.

Our solution strategy involves dividing the problem into subproblems, with each subproblem specified by a time index and an agent. The agents are numbered in a pre-specified order. We may define the following terms:

$Q =$ The set of subproblems.

$A_q =$ The subset of attribute space A for subproblem q.

$R_q =$ The state of the system corresponding to subproblem q

$= (R_a)_{a \in A_q}$

The state of the system is altered by two kinds of information processes - exogenous and endogenous. Exogenous information refers to that which arrives from outside the system. This could simply be the number of new resources entering the system. We use $_a$ to denote the number of resources with attribute a that enter the system. $_q$ would denote the vector of resources of different types entering subproblem q.

Endogenous information refers to decisions to act on the resources, which can change the attributes of the resources. We let $_q$ denote the set of decisions available to subproblem q. We also define the following:

$c_{ad} =$ The contribution from applying decision d on resource with attribute vector a.

$x_{ad} =$ The number of times decision d is used to act on a resource with attribute vector a.

$x_q =$ The vector of decisions taken in subproblem q,

$= (x_{ad})_{a \in A_q, d \in D_q}$.
We also define $C_q(R_q, x_q)$ as the contribution func-

tion that captures the total rewards from implementing the decisions in subproblem q, $C_q(R_q,x_q) = \sum_{a\in A_q}\sum_{d\in D_q} c_{ad}x_{ad}$.
It is useful to have notation that expresses the modification undergone by a resource that is acted upon by a decision. We do this using, $a' = a^M(a,d)$. We refer to a^M as the *attribute transition function*. We may also define the indicator function,

$$\delta_{a'}(a,d) = \begin{cases} 1 & if a' = a^M(a,d), \\ & 0 \text{ otherwise.} \end{cases}$$

In this context, it is useful to give formal definitions of the forward-reachable set ($\overrightarrow{M}_q$) and the backward-reachable set ($\overleftarrow{M}_q$) as described in [45]: $\overrightarrow{M}_q = \{q' \in |\exists\, a \in_q, d \in_q\}$ Such that $a' \in_{q'}$ and where, $a' = a^M(a,d)$ and q' is a reachable subproblem. Indeed, we will have $\overleftarrow{M}_q = \left\{q' \in |q \in \overrightarrow{M}_{q'}\right\}$.

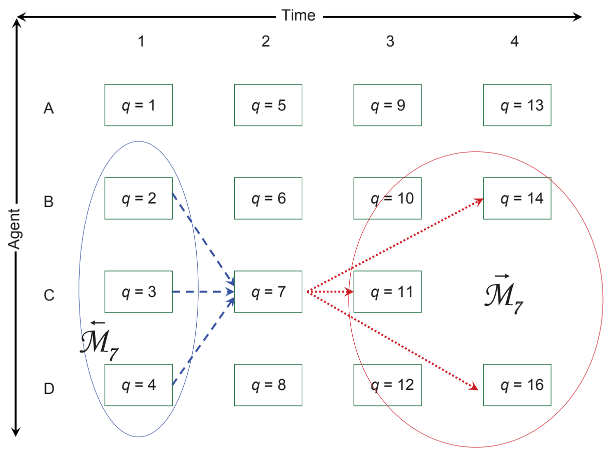

Figure 1. Illustration of forward- and backward-reachable sets.

In figure 1, we illustrate the idea of forward-reachable and backward-reachable sets. For example, as per the illustration, decisions taken on resources in subproblems 2, 3 and 4 alter their attributes so as to produce resources with attributes corresponding to subproblem 7. Hence, those subproblems (2, 3 and 4) would constitute the backward-reachable set, $\overleftarrow{M}_7$, with respect to subproblem 7. Similarly, decisions used upon resources in subproblem 7 transform them to resources belonging to subproblems 11, 14 and 16, which comprise the forward-reachable set corresponding to subproblem 7.

Using the notion of the forward-reachable set, we may further define,

The vector of decisions taken in subproblem q that results in resources with attributes corresponding to subproblem $q' \in \overrightarrow{M}_q$,

$(x_{ad})_{a\in_q, d\in_q}$ such that $a' \in_{q'}$ where $a' = a^M(a,d)$.

$\left(x_{qq'}\right)_{q'\in\overrightarrow{M}_q}$.

We denote by $_q$ the set of feasible decisions that have to satisfy the following constraints:
$\sum_{d\in_q} x_{ad} = R_a \;\; \forall a \in_q$,
$\sum_{a\in_q} x_{ad} \geq R^D_{b_d} \;\; \forall d \in^D_q$,
$x_{ad} \geq 0 \;\; \forall a \in_q, d \in_q$. Equation (2) is the flow conservation constraint for resources. Equation (2) constrains the system so that each demand is covered only once. Here, ^D_q is the set of decisions (in subproblem q) to cover a demand such that for each $d \in^D$, there exists a demand attribute vector $b_d \in_q$, where $_q$ is the set of demand attribute state vectors. $R^D_{b_d}$ denotes the number of demands with attribute vector b_d.

Next, we introduce the notion of a policy π which is a rule for choosing a decision given the information available. We use Π to denote the set of all policies. Decisions are made using a ***decision function*** which we represent using $X^\pi_q(R_q)$ and which is a function returning a feasible decision vector x_q under a policy π.

We typically have a family of decision functions, where for $\pi \in \Pi$, X^π_q is a particular decision function. Our challenge is to choose the best decision function.

Our optimization problem involves finding the best decision function $X^\pi_q(R_q)$ that solves $\max_{\pi\in\Pi} \mathbb{E}\left\{\sum_{q\in} C_q\left(R_q, X^\pi_q(R_q)\right)\right\}$. A myopic policy would solve this problem by stepping through the series of subproblems, $q \in$, optimizing over the current contributions alone.

3 Categories of information

Allowing information sharing between agents in a multi-agent system brings into the picture the cost of communication and reliability. The extent to which information is shared between agents is determined by the benefits obtained by each agent as a result. It also depends on the system architecture. The multi-agent system could be cooperative, where the agents share a common goal, or semi-cooperative, where each agent sees its own rewards. Communication between agents could resolve conflicts that might arise in situations where resources controlled by different agents are substitutes of each other, for completing a certain task. For example, a task could be assigned to the resource that is physically nearest to it. In another setting, different agents control resources that are complements in that each resource contributes a different quality or skill necessary to complete a particular task. In such a situation, communication becomes essential. In this section, we categorize the information that could be communicated between agents.

3.1 Knowledge

Knowledge refers to information in the form of data such as the resource classes and their attributes, parameters that govern the physics of the system (for example, costs associated with an action) or functional knowledge which includes functions that enable the decision-maker to make inferences about uncertain quantities.

The different agents in the system may not necessarily share information, but they could have knowledge about the resources belonging to other agents and the actions that they are capable of taking [51]. Other work such as [52], [53] and [54] also use prior knowledge to coordinate agents in the systems. Examples of forms of knowledge communicated include sensory information [29],[8] and resource availabilities [28]. The paper [31]describes a system where there is explicit communication of commands, parameter setup and assignment of tasks to agents involved in a search operation. The agents communicate with each other regarding formation of teams to undertake the task collectively. Other search and surveillance procedures involve communication of search routes [32], locations of observed targets [30], [29] and results of sensor actions in addition to UAV locations [55].

3.2 Forecasts of exogenous information processes

These are endogenously created forecasts of exogenous events. We use Ω to denote the set of these forecasts. Classic deterministic rolling horizon models use point forecasts (where $|\Omega| = 1$), whereas stochastic optimization models make use of distributional forecasts. One agent can communicate its own forecasts to another agent. For example, one agent may tell another agent that they think a storm will be developing later in the day, or that they think an aircraft might finish its repairs and be ready to fly.

3.3 Forecasts of the impact of decisions on the future

There are multiple approaches to quantify the impact of decisions made in one subproblem on another. In stochastic programming, the impact of a decision now on the future is captured using recourse functions, which are usually approximated using Bender's cuts [56], [57], [58], [59]. Dynamic programming uses value functions that are computed either using classical backward techniques (e.g. [60]) or using adaptive methods based on Monte Carlo sampling in approximate dynamic programming ([61], [62], [63]). There is a vast range of approximation strategies that can be used in approximate dynamic programming, effectively drawing on the entire field of statistics [64] and machine learning. In this paper, we are particularly interested in problems that involve the management of resources, which lends itself to approximation strategies that take advantage of the natural concavity of the value function. This property has been exploited extensively in the stochastic programming literature [65], and numerous publications in the operations research literature on topics such as inventory planning and transportation (see chapters 11 and 12 of [63] for an overview).

3.4 Forecasts of future decisions

Forecasts of decisions can be classified into three types:

1. Plans - These refer to explicit forecasts of future decisions at some level of aggregation. The agents have to make detailed decisions to match the plan, possibly as a penalty term (not a hard constraint).

2. Patterns - These are forecasts of future decisions based on the historical behavior of past decisions. Patterns are usually specified at some level of aggregation [66].

3. Policies - These are rules that govern how decisions should be made in the future. Policies are explicit mappings of states to actions at the most detailed levels.

Agents can communicate what they expect to do (a plan), or information that allows others to anticipate their behavior.

Papers [10], [11] and [12] describe problems where multiple agents share the same policies, however, their behaviors may differ because of different inputs. In [20], though there is no explicit communication, agents see the outcome of each other's actions. The paper [53] describes a procedure where each agent learns the frequencies of actions (in effect, policies) of other agents by observing their actions and uses that as a forecast of policies of the other agents.

3.5 History

This refers to the information regarding the actual sequence of events which have already happened, and which an agent may (partially or fully) communicate with other agents. Examples include the results of a mission, the movements of a plane that occurred yesterday, or the amount of fuel that was used.

Historical data is used in a feedback loop to update system knowledge. For example, knowing that a mission was successful helps an agent update the probability of success. Knowing the amount of fuel that was used allows an agent to update his estimate of how much fuel is required for a mission. A demand forecasting model (which constitutes functional knowledge) is updated using the observed demand data from previous time-periods. In the case of semi-cooperative multi-agent systems, a significant portion of historical data could be communicated to an agent by other agents such as partial results of previous decisions (as in [67]) or actual decisions ([68], where agents communicate speech acts). In such a situation, there could be distortion involved in the communication due to transmission noise. Another possibility is that the distortion is intentional on the part of the communicating agent. For example, an agent that normally has an excess of demands to serve during a certain period may inflate its need for extra resources to another agent.

4 Solution strategy

Our solution strategy involves formulating the problem as a dynamic program. Classical models of dynamic programs [60], centered around Bellman's optimality equation, can be solved to optimality using standard techniques. However, such techniques encounter the classic curse of dimensionality.

4.1 Approximate dynamic programming

Our approach to overcome this is to first formulate the post-decision state of the system, $R^x_{qq'}$, resulting from implementing the decision vector, $x_{qq'}$, in subproblem q. $\mathrm{R}^x_{qq'a'} = \sum_{a\in_q}\sum_{d\in_q} x_{ad}\delta_{a'}(a,d)\ \forall q' \in \overrightarrow{M}_q,\ a' \in_{q'}$, $R^x_{qq'} = (R^x_{qq'a'})_{a'\in_{q'}}\ \ \forall q' \in \overrightarrow{M}_q$. The post-decision state is related to the pre-decision state, R_q, by way of the following equation, $\mathrm{R}_q = \sum_{q'\in\overleftarrow{M}_{q+1}} R^x_{q',q} +_q$ where $_q$ denotes the exogenous changes to the resource state in subproblem q and $R^x_{q',q}$ denotes the pre-decision state of the system resulting from the decision vector, $x_{q',q}$.

In order to capture the impact of the decisions applied in subproblem q on future subproblems, we define a value function associated with this change. We let $V_{qq'}(R^x_{qq'})$ denote the value of being in post-decision state, $R^x_{qq'}$.

The next step is to replace the function $V_{qq'}(R^x_{qq'}s)$ with an approximation. There are a number of strategies we can use. For our work, we used $\mathrm{V}_{qq'}(R^x_{qq'}) =_{qq'} (R^x_{qq'})$ $= \sum_{a\in\ qq'a}(R^x_{qq'a})$, where $_{qq'a}(R_{qq'a})$ is piecewise linear and concave in $R_{qq'a}$. This approximation has worked well in a number of fleet management problems [69], [70], [71], [48].

Figure 2 depicts the use of value function approximations to to capture the impact of decisions applied to resources in the current subproblem on future subproblems.

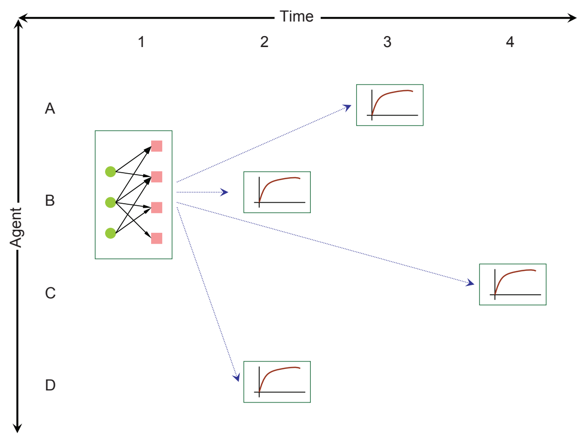

Figure 2. Illustration of future impact of decisions. Implementing decisions on resources in the current subproblem has an impact on future subproblems, that is captured using value function approximations.

The approximation of the value function is done using an iterative procedure. We let n denote the iteration counter. At iteration n, we solve the decision problem given by $x_q^n(R_q^n) = \arg\max_{x_q \in_q} \left[C_q(R_q^n, x_q) + \sum_{q \in \overrightarrow{M}_q} {}_{qq'}(R_{qq'}^x) \right]$ subject to the constraints in equations (2) - (2). For this specific application, equation (4.1) is an integer program which can be solved using a commercial solver. Let ${}_q(R_q^n)$ be the objective function value obtained by solving (4.1). We can update our value function using estimates of the derivative of ${}_q(R_q)$ with respect to R_q. The updating process can be represented using the equation, ${}_{t-1}^{n}(R_{t-1}^x) = U^V({}_{t-1}^{n-1}(R_{t-1}^x), R_{t-1}^{x,n}{}_t^n)$.

4.2 The basic algorithm

The algorithmic strategy for solving the multi-agent problem is outlined in figure 3. We start with some initial approximation of the value functions for the various states. We then iteratively step through the subproblems and solve each one based on the current approximations of the value functions. We define the marginal value of a resource as the value of having an additional unit of a resource of that particular type in the system. The solutions to each subproblem give a set of marginal values, which are then used to update the value function approximations of agent q and also other agents in the backward reachable set $\overleftarrow{M}_q$. We terminate the simulation based on some stopping criterion, such as some margin of improvement of the overall objective function, or, as depicted in the algorithm, at the end of a certain number (N, in this case) of iterations. The value function approximations returned at the end of the simulation can be used to derive an appropriate policy (which specifies what action to take in each state) to be used while solving each subproblem.

4.3 Communication with distortion

In practical situations, the information that is communicated between agents could be biased because an agent chooses not to share it exactly with other agents. There could also be noise in the communication due to errors in the measurement or transmission.

In the multi-agent model described in section 2, information is shared between subproblems in the form of marginal values $(\pi_{qq'}^n)$ to update the value function approximations $\left\{{}_{qq'}(.)\right\}_{q,q'}$ (see equation (4.1)). In an idealized situation, the marginal values are communicated without distortion, as in Step 3 of Figure 3.

In semi-cooperative systems, when the information is shared between subproblems controlled by different agents, it may well be the case that it is distorted. For example, if the agent in subproblem q' has an excess of demands and requires resources of type a, it may choose to inflate the marginal values $(\pi_{qq'a})$. We represent the distorted value that is communicated to subproblem q as, $\overrightarrow{\pi}_{qq'a} = \pi_{qq'a} + \varepsilon_{qq'a}$ $\varepsilon_{qq'a}$ is called noise if its mean is zero. A non-zero mean value of distortion is termed communication bias.

5 Experimental Results

The central questions that we attempt to answer using experiments are: 1.) how well can a semi-cooperative framework match a system optimal with the right communication?, and 2.) what is the effect of communication noise and biases? With these questions in mind, we design experiments with 3 agents, 100 locations, 10 resources and 100 demands arising over a simulation horizon of 12 hours. Each agent needs to use its resources

```
Step 0. Initialization:
        Step 0a. Initialize $\left\{{}^{0}_{q}(R_q)\right\}, q \in$ .
        Step 0b. Set $n = 1$.
        Step 0c. Set the starting state - $R^1_0$.
Step 1. For $q = 1,2,\ldots,||$:
Step 1a. Solve, $x^n_q = \arg\max_{x_q \in_q} \left[ C(R^n_q, x_q) + \sum^{n-1}_{q' \in \overrightarrow{M}_q} {}_{qq'} \left( R^x_{qq'} \right) \right], subjectto:$
        $\sum_{d \in_q} x_{ad} = R_a \;\; \forall a \in_q, x_{ad} \geq 0 \;\; \forall a \in_q, d \in_q, where, foreach$ q' $\in \overrightarrow{M}_q$, the elements of $R^x_{qq'}$ are computed using, $R^x_{qq'a'} = \sum_{a \in_q} \sum_{d \in_q} x_{ad} \delta_{a'}(a,d) \;\; \forall\, a' \in_{q'}$ .
Step 1b. Obtain the dual variables $\{\pi^n_{qq'\, q' \in \overrightarrow{M}_q \bigcup \{q\}}\}$ from constraints (4.2) and (4.2).
Step 1c. Update the state for the subsequent subproblem:
        $R^n_{q+1} = \sum_{q' \in \overleftarrow{M}_{q+1}} R^{x,n}_{q',q+1} + {}^n_{q+1}, where R^{x,n}_{q',q+1} = \left( R^{x,n}_{q',q+1,a} \right)_{a \in_{q+1}}$ and $R^{x,n}_{q',q+1,a}$ is obtained using a similar expression as in equation (4.2).
Step 2. For $q = 1,2,\ldots,||$ , define the updating vector, ${}^n_q = \left( \pi^n_{qq'} \right)_{q' \in \overrightarrow{M}_q}$ .
        Update the value function approximations with appropriately chosen stepsizes, $\alpha^n$, ${}^n_q(R_q) = (1-\alpha^n){}^{n-1}_q(R_q) + \alpha^{n n}_q \;\; \forall q \in$ .
Step 3. Let $n = n+1$. If $n \leq N$, go to step 1.
Step 4. Return the value function estimates, $\left\{{}^n_q(R_q)\right\}_{q \in}$.
```

Figure 3. An ADP algorithm for multi-agent systems.

to serve the demands, within a certain time window after which the demands expire. A reward is obtained when a demand is served. The resources and demands are characterized by types. A demand of a particular type has to be served by a resource of a certain type, otherwise a substitution cost is incurred. There is also a cost incurred if a demand, after arrival, has to wait for multiple time periods before being served.

The resource attribute vector has the following dynamic elements - current time, current location and controlling agent, as well as the static elements of resource-type and ownership (which refers to the agent that owns the resource). Demands have the attributes of time, location, expiration time, type and ownership of which time is the only dynamic attribute. Demands disappear from the system if they are not served before their expiration time. Figure 4 gives an illustration of demands arising at different points in time over the simulation horizon. Each column represents the number of demands, colored by agent, that arise during that time period. The distribution of demands over time among the three agents is such that in almost every time period there is always some agent with an excess of demands. In this setting, semi-cooperative sharing of resources between the agents could improve the overall demand coverage.

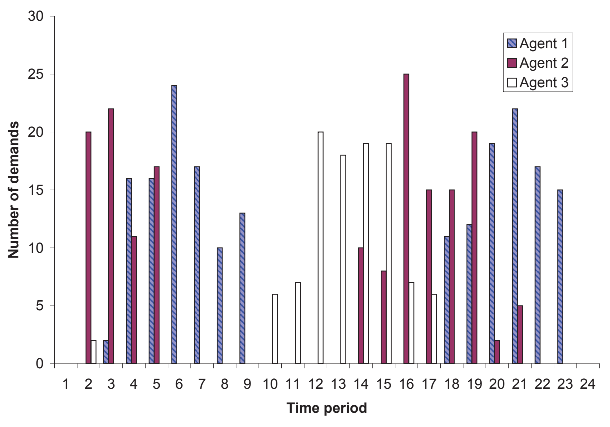

Figure 4. Distribution of demands in a 3-agent system. Demands are unequally distributed among the agents in different time periods, creating a need for resource-sharing or demand-sharing among the agents.

In each subproblem, the following list of decisions are available for the decision-maker:

– Hold resource or demand - Both resources and demand can be held at their current location. Holding a demand results in a cost that is proportional to the time it is held since its arrival at the location. Furthermore, a demand can be held only until it expires.

– Move resource - Resources can be moved to other locations at a cost, where demands may be more likely to arise.

– Serve demand - Resources can be used to serve demands, which results in a reward. If there is a type mismatch between the resource and the demand, a substitution cost is incurred. The other costs incurred with this decision include the transportation cost of moving the resource to the demand location as well as the waiting cost for the demand which is proportional to the time it has to wait before being served.

– Transfer control - Agents may choose to borrow resources of another agent while incurring a cost, in the event of a deficit of resources under their control to serve demands.

5.1 Value of communication

In this section, we evaluate the benefits of communication between agents. In the semi-cooperative system, the communication is in the form of marginal values ($\pi_{qq'}$) of resources as outlined in Section 4.

Figures 5 and 6 contrast the behavior with and without communication in a 2-agent system. In this configuration, agent 1 has an excess of demands while agent 2 has a surplus of resources, most of which remain under-utilized in the absence of communication (Figure 5). In the semi-cooperative system, agent 1 communicates the marginal values of resources to agent 2 which causes the latter to lend its resources to the former, thereby resulting in greater demand coverage as depicted in Figure 6.

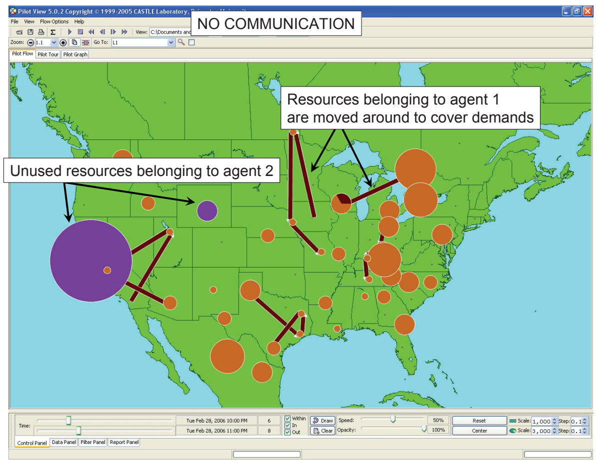
NO COMMUNICATION
Resources belonging to agent 1 are moved around to cover demands
Unused resources belonging to agent 2

Figure 5. A snapshot of the 2-agent system for a particular time period. In this scenario, there is no communication between agents. Much of the resources remain unutilized as a result of lack of cooperative behavior.

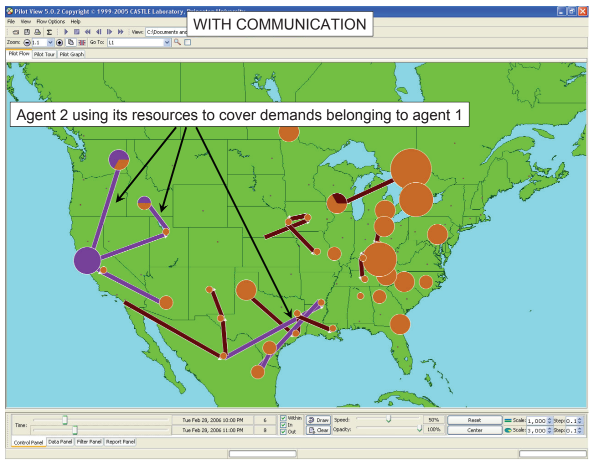
WITH COMMUNICATION
Agent 2 using its resources to cover demands belonging to agent 1

Figure 6. The 2-agent system with cooperative behavior. The resources that were previously unutilized are now being used to cover the excess demands.

In Figure 7, we analyze the percentage of demands successfully served by multi-agent system architectures with and without communication, as a function of the number of iterations of the algorithm in Figure 3. As a benchmark, we also perform the tests on a single agent system with the same number of resources. As is evident from the graph, allowing for communication (and hence, transfer of resources), between the agents allows for a much better coverage of the demands, matching that of the single agent system.

Table 1. Percentage of demands served for several problem instances. *TW* and *LC* denote parameter settings. The remaining columns represent the competing system architectures - SA (Single agent), MAC (Multi-agent semi-cooperative) and MANC (Multi-agent with no communication)

No.	*Agents*	*TW*	*LC*	SA	MAC	MANC
1	*2*	*No*	*Yes*	19	17	9
2	*2*	*Yes*	*Yes*	31	29	14
3	*2*	*No*	*No*	66	68	39
4	*2*	*Yes*	*No*	90	78	72
5	*2*	*Yes*	*No*	94	76	66
6	*3*	*No*	*Yes*	26	22	15
7	*3*	*Yes*	*Yes*	63	51	37
8	*3*	*No*	*No*	98	90	68
9	*3*	*Yes*	*No*	100	96	87

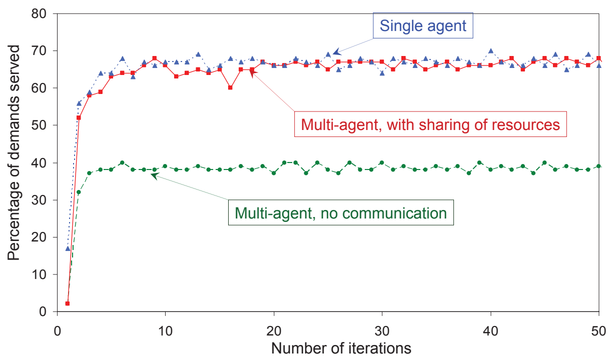

Figure 7. Percentage of demands served for different system architectures.

Table 1 provides a summary of the percentage of demands served for different problem instances. We varied three parameter settings to generate the different instances. The first one was the number of agents in the system. The second setting, *TW* (time-windows), involved whether or not to allow the demands to be held for more than a single time-period. The final setting, *LC*, denotes whether or not we impose a location constraint where a resource could serve a demand only if it is in the same location. In all the problem instances, the semi-cooperative model with communication consistently outperforms the multi-agent system with no communication and also approaches the single agent architecture in terms of performance.

5.2 Effect of distortion

In this section, we investigate the effect of distortion in the information that is communicated between the agents in a semi-cooperative model. The marginal values of resources are inflated and noise is added before they are communicated to other agents. As expected, this results in a solution which is poorer in quality as compared to the case where there is no distortion in the communication. However, the semi-cooperative system with distorted communication still outperforms the system with no communication between the agents, as illustrated in Figure 8.

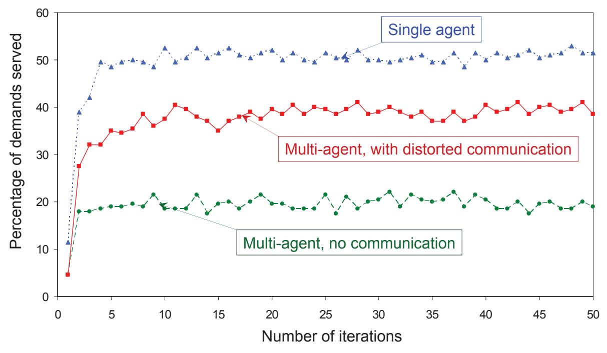

Figure 8. Effect of distortion in information communicated.

6 Conclusions

We have developed a multi-agent system where the agents are autonomous, but demonstrate semi-cooperative behavior to achieve a common objective of maximizing demand coverage. Each agent possesses ownership of a certain number of resources. During the course of the simulation, information regarding the future value of resources is communicated between the agents. This is used by the agents to determine whether or not to transfer control of under-utilized resources to other agents, thereby maximizing the overall number of demands served by the system. Our experimental results show that this type of semi-cooperative behavior improves the objective function compared to a system where the agents act independently with no cooperation. With no distortion in the values communicated, the system performance approaches that of a system where there is a single agent controlling all the resources. Even with distorted information being communicated, the semi-cooperative system still outperforms the system with independent agents. The advantage that the semi-cooperative system has over the single agent system is that the decision problems are much smaller in size due to decentralization, but we are still able to obtain results that are near-optimal.

References

[1] J. Marschak, Elements for a theory of teams, Management Sci., vol. 1, pp. 127137, 1955.

[2] R. Radner, Team decision problems, Ann. Math. Stat., vol. 33, pp. 857881, 1962.

[3] R. Beckers, O. E. Holland, and J. L. Deneubourg, From local actions to global tasks: Stigmergy in collective robotics, in Artificial Life IV, R. Brooks and P. Maes, Eds. Cambridge, MA: MIT Press, 1994, pp. 181189.

[4] R. Arkin, Cooperation without communication: Multi-agent schemabased robot navigation, Journal of Robotic Systems, vol. 9, no. 3, pp. 351364, 1992.

[5] L. Steels, A case study in the behavior-oriented design of autonomous agents, in Proc. Simulation of Adaptive Behavior, 1994.

[6] P. Stone and M. Veloso, Multiagent systems: A survey from a machine learning perspective, Autonomous Robots, vol. 8, pp. 345383, 2000.

[7] M. Wiering, B. Krose, and F. C. A. Groen, Learning in multi-agent systems, University of Amsterdam, Amsterdam, The Netherlands, Tech. Rep., 2000.

[8] P. Davidsson, L. Henesey, L. Ramstedt, J. Tornquist, and F. Wernstedt, An analysis of agent-based approaches to transport logistics, Transportation Research, vol. C13, pp. 255271, 2005.

[9] M. J. Mataric, Using communication to reduce locality in multirobot learning, in Proceedings of the National Conference on Artificial Intelligence, 1997, pp. 643648.

[10] R. Salustowicz, M. Wiering, and J. Schmidhuber, Learning team strategies: Soccer case studies, Machine Learning, vol. 33, no. 2-3, pp. 263282, 1998.

[11] M. A. Wiering, R. P. Salustowicz, and J. Schmidhuber, Cmac models learn to play soccer, in Proceedings of the 8th International Conference on Artificial Neural Networks (ICANN98), M. B. L. Niklasson and T. Ziemke, Eds., vol. 1. Springer-Verlag, London, 1998, pp. 443448.

[12] M. Dorigo and G. Di Caro, The ant colony optimization metaheuristic, in New Ideas in Optimization, D. Corne, M. Dorigo, and F. Glover, Eds. London: McGraw-Hill, 1999, pp. 1132.

[13] J. Schneider, W.-K. Wong, A. Moore, and M. Riedmiller, Distributed value functions, in Proc. 16th International Conf. on Machine Learning. Morgan Kaufmann, San Francisco, CA, 1999, pp. 371378.

[14] R. Crites and A. Barto, Elevator group control using multiple reinforcement learning agents, Machine Learning, vol. 33, pp. 235262, 1994.

[15] M. Littman and J. Boyan, A distributed reinforcement learning scheme for network routing, in Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications, J. Alspector, R. Goodman, and T. X. Brown, Eds., 1993, pp. 4551.

[16] Y. Ye and J. K. Tsotsos, On the collaborative object search team: A formulation, in DAI Meets Machine Learning, Lecture Notes in Artificial Intelligence, G. Weiss, Ed. Springer-Verlag, 1997, pp. 94116.

[17] N. Carver, Z. Cvetanovic, and V. R. Lesser, Sophisticated cooperation in FA/C distributed problem solving systems, in Proceedings of the Ninth National Conference on Artificial Intelligence, 1991.

[18] G. Zlotkin and J. S. Rosenschein, Mechanisms for automated negotiation in state oriented domains, Journal of Artificial Intelligence Research, vol. 5, pp. 163238, 1996.

[19] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, Cooperative mobile robotics: Antecedents and directions, Autonomous Robots, vol. 4, pp. 727, 1997.

[20] S. Sen, M. Sekaran, and J. Hale, Learning to coordinate without sharing information, in Proceedings of the Twelfth National Conference on Artificial Intelligence, Seattle, WA, 1994, pp. 426431.

[21] J. Mayer, Stochastic linear programming algorithms: A comparison based on a model management system. Springer, 1998.

[22] M. J. Shaw and R. Sikora, A distributed problem-solving approach to rule induction: Learning in distributed artificial intelligence systems, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep., November 1990.

[23] S. S. Sian, Adaptation based on cooperative learning in multi-agent systems, in Decentralized A.I. 2: Proceedings of the 2nd European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Y. Demazeau and J. Muller, Eds. Amsterdam: North-Holland, 1991, pp. 257272.

[24] S. D. Whitehead, A complexity analysis of cooperative mechanisms in reinforcement learning. in Proceedings of the Ninth National Conference on Artificial Intelligence, 1991, pp. 607613.

[25] M. Tan, Cost-sensitive reinforcement learning for adaptive classification and control. in AAAI, 1991, pp. 774780.

[26] G. Wei, Learning to coordinate actions in multi-agent systems, in Proceedings of the International Joint Conference on Artificial Intelligence, August 1993, pp. 311316.

[27] R. H. Crites and A. G. Barto, Improving elevator performance using reinforcement learning, in Advances in Neural Information Processing Systems, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds., vol. 8. The MIT Press, 1996, pp. 10171023.

[28] R. G. Smith, The contract net protocol: High-level communication and control in a distributed problem solver, IEEE Transactions on Computers, vol. C-29, no. 12, pp. 11041113, 1980.

[29] M. Flint, E. Fernandez, and M. Polycarpou, Stochastic models of a cooperative autonomous uav search problem, Military Operations Research, vol. 8, no. 4, pp. 1332, 2003.

[30] M.-W. Jang, S. Reddy, P. Tosic, L. Chen, and G. Agha, An actorbased simulation for studying uav coordination, Proceedings of the 15th European Simulation Symposium, 2003.

[31] Y. Dongyong, T. Qiong, F. Luping, Z. Xiao, and J. Jingping, Cooperative multi-agent transport based on coevolution, August 2004.

[32] P. B. Sujit and D. Ghose, Search using multiple uavs with flight time constraints, IEEE Transactions on Aerospace and Electronic Systems, vol. 40, no. 2, pp. 407463, April 2004.

[33] E. H. Durfee, V. R. Lesser, and D. D. Corkill, Distributed artificial intelligence, in Cooperation Through Communication in a Distributed Problem Solving-Network, M. N. Huhns, Ed. Los Altos, CA: Morgan Kauffman, 1987, p. 2958.

[34] R. Pope, S. Conry, and R. Meyer, Distributing the planning process in a dynamic environment, in Proceedings of the Eleventh International Workshop on Distributed Artificial Intelligence, Glen Arbor, MI, 1992, p. 317331.

[35] D. D. Corkill, A framework for organizational self-design in distributed problem solving networks. Ph.D. dissertation, University of Massachusetts, Amherst, MA., 1982.

[36] L. Gasser and M. N. Huhns, Distributed artificial intelligence. San Mateo, CA, USA: Morgan Kaufmann Publishers Inc., 1989, vol. 2.

[37] M. Tan and R. Weihmayer, Integrating agent-oriented programming and planning for cooperative problem-solving, in Proceedings of the AAAI-92s Workshop on Cooperation among Heterogeneous Intelligent Agents, San Jose, MI, 1992.

[38] M. P. Wellman, A market-oriented programming environment and its application to distributed multicommodity flow problems, Journal of Intelligence Research, vol. 1, pp. 123, 1993.

[39] G. Dantzig and P. Wolfe, Decomposition principle for linear programs, Operations Research, vol. 8, pp. 101111, 1960.

[40] W. J. Baumol and T. Fabiani, Decomposition, pricing for decentralization and external economies, Management Science, vol. 11, no. 1, pp. 132, September 1964.

[41] A. A. Assad, Multicommodity network flows-A survey, Networks, vol. 8, pp. 3791, 1978.

[42] J. F. Kurose and R. Simha, A microeconomic approach to optimal resource allocation in distributed computer systems, IEEE Transactions on Computers, vol. 38, no. 5, pp. 705717, 1989.

[43] V. R. Lesser, Cooperative multiagent systems: A personal view of the state of the art, IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 1, pp. 133142, January/February 1999.

[44] H. Topaloglu and W. B. Powell, Dynamic programming approximations for stochastic, time-staged integer multicommodity flow problems, Informs Journal on Computing, vol. 18, no. 1, pp. 3142, 2006.

[45] J. A. Shapiro and W. B. Powell, A metastrategy for large-scale resource management based on informational decomposition, Informs Journal on Computing, vol. 18, no. 1, pp. 4360, 2006.

[46] W. B. Powell, J. A. Shapiro, and H. P. Simao, A representational paradigm for dynamic resource transformation problems, in Annals of Operations Research, R. F. C. Coullard and J. H. Owens, Eds. J.C. Baltzer AG, 2001, pp. 231279.

[47] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, Eds., Handbook of Learning and Approximate Dynamic Programming. New York: IEEE Press, 2004.

[48] T. Wu, W. Powell, and A. Whisman, The optimizing simulator: An intelligent analysis tool for the military airlift problem, Princeton University, Princeton, NJ, Tech. Rep., 2007.

[49] G. Godfrey and W. B. Powell, An adaptive, dynamic programming algorithm for stochastic resource allocation problems II: Multi-period travel times, Transportation Science, vol. 36, no. 1, pp. 4054, 2002.

[50] W. B. Powell and H. Topaloglu, Stochastic programming in transportation and logistics, in Handbooks in Operations Research and Management Science: Stochastic Programming, A. Ruszczynski and A. Shapiro, Eds., vol. 10. Amsterdam: Elsevier, 2003, pp. 555635.

[51] M. S. Fox, An organizational view of distributed systems, in Distributed Artificial Intelligence. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[52] P. Brazdil, M. Gams, S. Sian, L. Torgo, and W. van de Velde, Learning in distributed systems and multi-agent environments, in Lecture Notes in Artificial Intelligence Vol.482: Machine Learning-EWSL-91, Y. Kodratoff, Ed., 1991, pp. 412423.

[53] C. Claus and C. Boutilier, The dynamics of reinforcement learning in cooperative multiagent systems, in Proceedings of the Fifteenth National Conference on Artificial Intelligence, 1998, pp. 746752.

[54] S. Bergman, E. Pavlov, and J. S. Rosenschein, Negotiation in stateoriented domains with incomplete information over goals, in European Conference on Artificial Intelligence, 2004, pp. 812.

[55] M. Flint and E. Fernandez, Approximate dynamic programming methods for cooperative uav search, BAE Systems, Advanced Information Technologies, Burlington, MA, Tech. Rep., 2005.

[56] R. Van Slyke and R. Wets, L-shaped linear programs with applications to optimal control and stochastic programming, SIAM Journal of Applied Mathematics, vol. 17, no. 4, pp. 638663, 1969.

[57] J. Birge, Decomposition and partitioning techniques for multistage stochastic linear programs, Operations Research, vol. 33, no. 5, pp. 9891007, 1985.

[58] J. Higle and S. Sen, Stochastic decomposition: An algorithm for two stage linear programs with recourse, Mathematics of Operations Research, vol. 16, no. 3, pp. 650669, 1991.

[59] Z.-L. Chen and W. B. Powell, A convergent cutting-plane and partialsampling algorithm for multistage stochastic linear programs with recourse, Journal of Optimization Theory and Applications, vol. 102, no. 3, pp. 497524, 1999.

[60] M. L. Puterman, Markov Decision Processes. New York: John Wiley Sons, 1994.

[61] D. Bertsekas and J. Tsitsiklis, Neuro-Dynamic Programming. Belmont, MA: Athena Scientific, 1996.

[62] R. Sutton and A. Barto, Reinforcement Learning. Cambridge, Massachusetts: The MIT Press, 1998.

[63] W. B. Powell, Approximate Dynamic Programming: Solving the curses of dimensionality. New York: John Wiley and Sons, 2007.

[64] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. New York, NY: Springer series in Statistics, 2001.

[65] J. Birge and F. Louveaux, Introduction to Stochastic Programming. New York: Springer-Verlag, 1997.

[66] A. Marar, W. B. Powell, and S. Kulkarni, Capturing expert knowledge in resource allocation problems through low-dimensional patterns, IIE Transactions, vol. 38, no. 2, pp. 159172, 2006.

[67] E. H. Durfee and V. R. Lesser, Partial global planning: A coordination framework for distributed hypothesis formation, IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, no. 5, pp. 11671183, 1991.

[68] P. Cohen and C. R. Perrault, Elements of a plan-based theory of speech acts, Cognitive Science, vol. 3, no. 3, pp. 177212, 1979.

[69] G. Godfrey and W. B. Powell, An adaptive, dynamic programming algorithm for stochastic resource allocation problems I: Single period travel times, Transportation Science, vol. 36, no. 1, pp. 2139, 2002.

[70] W. B. Powell and H. Topaloglu, Fleet management, in Applications of Stochastic Programming, S. Wallace and W. Ziemba, Eds. Philadelphia: Math Programming Society - SIAM Series in Optimization, 2005, pp. 185216.

[71] H. Topaloglu and W. B. Powell, Dynamic programming approximations for stochastic, time-staged integer multicommodity flow problems, Informs Journal on Computing, vol. 18, no. 1, pp. 3142, 2006.