Distributed Task Assignment for Mobile Agents

Brandon J. Moore and Kevin M. Passino

***Abstract*—This note demonstrates how the distributed auction algorithm can be modified to assign mobile agents to spatially distributed tasks despite communication delays and the fact that agent movement may cause the benefit associated with each possible agent-task assignment to vary during the execution of the algorithm. Bounds on the convergence time of the algorithm and the sub-optimality of the resulting solution are provided. Monte Carlo simulations are provided to show the conditions under which the modified distributed auction can outperform centralized calculation.**

***Index Terms*—Author, please supply your own keywords or send a blank e-mail to keywords@ieee.org to receive a list of suggested keywords.**

I. Introduction

This note addresses a cooperative control problem in which a group of mobile agents must assign themselves to a group of spatially distributed tasks on a one-to-one basis while attempting to maximize a certain collective benefit function. The value of this collective benefit function is taken to be the sum of the individual benefits associated with each agent-task pair in the final assignment, where each of these values may depend on the particular agent-task pairing and the time that it takes that agent to travel to and complete that task. If inter-agent communication is very fast, then this problem takes the form of the *assignment problem* from the field of combinatorial optimization (with agents as people and tasks as objects) and many algorithms of polynomial complexity exist to solve it [1]–[3]. However, if inter-agent communication is subject to significant delays then it will take a nontrivial amount of time for the agent group to come up with a final assignment because either the agents will have to transmit their benefit information (i.e., the value each task has to them and the time it will take to complete it) to a group leader in order to solve the problem in a centralized fashion, or they can solve it in a decentralized manner by some message passing algorithm. Because the collective benefit function depends on the time each task is completed, the time interval taken to solve the assignment problem in this situation will have a detrimental impact on the benefit received from the final assignment. In addition, if the mobile agents cannot remain stationary (e.g., if they are autonomous air vehicles) then the individual agents may travel away from what is to be their assigned task during this interval, which can further delay the completion of that task and degrade the benefit received.

Our choices in dealing with this problem are either to accept the loss of benefit imposed by the delay in reaching a final assignment or to try to deal with the fact that the individual agent-task benefits may vary over time in certain ways by developing an algorithm that attempts to minimize the loss of benefit that may occur. In this note we modify the distributed auction algorithm [1] to do this by guiding the agents' motion during the run-time of the algorithm. In addition to analyzing the conditions under which this modified algorithm is still guaranteed to converge to a solution in finite time, we explore both its worse-case bound on the lost benefit and the practical results from simulations for some autonomous air vehicle example scenarios (and compare these values to those achieved by a centralized algorithm under the same circumstances).

There are a number of cooperative control problems that involve variants of the assignment problem. In [4], for instance, the distributed sequential shortest augmenting path algorithm is modified to deal with the dynamic arrival of additional tasks, while in [5], multiple assignment problems with evolving subsets of the agents and tasks are iteratively solved to provide a heuristic method of dealing with coupled tasks (i.e., tasks which must be completed in a specific order).

Manuscript received January 19, 2005; revised March 7, 2006. Recommended by Associate Editor A. Bemporad. This work was supported by the AFRL/VA and AFOSR Collaborative Center of Control Science under Grant F33615-01-2-3154. The work of B. Moore was supported by a DAGSI Fellowship

The authors are with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: mooreb@ece.osu.edu; passino@ece.osu.edu)

Digital Object Identifier 10.1109/TAC.2007.894545

II. Modified Algorithm

A. Benefit Functions

Let m be the number of mobile agents and let n $(n \geq m)$ be the number of tasks. In our scenario agents are considered to be nonrenewable resources in that they may complete only one task (as might be the case in a "kamikaze" type attack by an autonomous munition). To complete a task an agent must "arrive" at that task, where arrival can be either more or less restrictive than mere co-location. For example, an agent may have to arrive at the task at a specific angle or it may merely have to come within a certain distance of the task's location. Upon completion of a task an agent will receive a benefit that depends on that agent-task pair (because the agents' ability to complete a particular task may vary) and also on the time it takes to complete the task.

Let $s_i(t)$ be a vector describing the state of agent i at time t (e.g., location, heading, speed, etc.) and we assume that the trajectory $s_i(t), t \geq 0$ satisfies some sort of vehicle dynamics described in the form of differential equations and associated constraints. Let the constant d_j be a vector associated with task j which represents the pertinent information necessary to describe the conditions that an agent must satisfy in order to complete that task. Unless stated otherwise, we will assume that if agent i can ever reach task j in a finite length of time, then it can always do so. Under this assumption, for each agent i that can reach task j, there exists a time optimal trajectory from any point $s_i(t)$ to that task which we denote by $\sigma[s_i(t), d_j]$ and its travel time by the metric $|\sigma[s_i(t), d_j]|$. We now define the benefit that agent i *could* receive by completing task j at time $t + |\sigma[s_i(t), d_j]|$ as

$$a_{ij}(t) = C_{ij} - B(t + |\sigma[s_i(t), d_j]|) \tag{1}$$

where C_{ij} is a constant particular to the pairing (i, j), B is constant across all such pairs and defines the relative importance of completing tasks quickly, and the quantity $t + |\sigma[s_i(t), d_j]|$ is the time of arrival of agent i if it follows its optimal trajectory to task j from time t onward. Since the tasks are stationary and because $|\sigma[s_i(t), d_j]|$ is a optimal trajectory, its value must satisfy

$$\begin{aligned} &|\sigma[s_i(t+\Delta t), d_j]| \geq |\sigma[s_i(t), d_j]| - \Delta t \\ &\qquad\qquad \forall t \geq 0 \qquad \forall 0 \leq \Delta t \leq |\sigma[s_i(t), d_j]| \end{aligned} \tag{2}$$

with equality if and only if agent i *tracks* task j from t to $t + \Delta t$ (i.e., follows the optimal trajectory to that task). For this work we will make the assumption that for any trajectory agent i might follow from time t to time $t + \Delta t$, the increase in the travel time of the optimal trajectory to a given task j can be bounded from above by

$$|\sigma[s_i(t+\Delta t), d_j]| \leq |\sigma[s_i(t), d_j]| + W + V\Delta t \qquad \forall t, \forall \Delta t \geq 0 \tag{3}$$

where the constants $W \geq 0$ and $V \geq -1$ are identical for every agent i and every task j. An inequality of this form should be satisfied for many situations with realistic vehicle dynamics (i.e., because agents can usually can move away from a task at the same speed they can move towards it, V will typically be equal to one and W will be related to restrictions imposed by non-holonomic constraints). For example, the constant speed Dubins Car model [6] frequently used to represent air vehicles satisfies this assumption with $V = 1$ and $W = (2+3\pi)/\omega_{\max}$, where $\omega_{\max}$ is the maximum turning rate in radians per second.

With the benefit function defined in (1), using the inequalities in (2) and (3) we can place the following bounds on a_{ij}:

$$a_{ij}(t+\Delta t) \geq a_{ij}(t) - B(W + (V+1)\Delta t)\ \forall\, t, \Delta t \geq 0 \qquad (4)$$
$$a_{ij}(t+\Delta t) \leq a_{ij}(t) \qquad \forall\, t \geq 0,\ \forall\, 0 \leq \Delta t \leq |\sigma[s_i(t), d_j]| \quad (5)$$

where equality is reached in (5) if and only if agent i tracks task j from time t to time $t+\Delta t$. The qualification in (5) stems from the fact that once an agent reaches a task, the benefit will start to decrease again if it does not complete that task. If the agent continues to track the task and is able to remain at the task location, $a_{ij}(t)$ will decrease at a rate of $B\Delta t$. If it cannot remain stationary, the agent must follow a loop back to the task, and so $a_{ij}(t)$ will drop by the length of that loop and then remain steady until the next time the agent reaches the task.

At this point let us denote t_a as the time a full assignment is reached and t_f^i as the time agent i completes its task. Since each agent will track its assigned task from t_a until t_f^i, it is clear from (5) that the collective benefit of the assignment at t_a is the same as what the agents will receive when they complete their tasks. Also, during the time the agents take to calculate and communicate in order to reach their eventual assignment, the collective benefit of that assignment is degrading (except for the unlikely case where every agent somehow happens to always track the task to which they will eventually be assigned). Given that fact, we are motivated to develop an algorithm that controls the motion of the agents during the time interval $[0, t_a]$ in a manner that seeks to reduce the degradation of the collective benefit. The modification of the distributed auction algorithm [1] proposed in the next section attempts to do just this.

B. Motion Control Algorithm

The distributed auction algorithm [1] involves persons placing bids on objects based on their current *price* and potential benefit (with the highest bidder for an object at a given stage temporarily assigned to that object). As the prices of the objects are increased by the bidding they become less and less attractive to the persons, until eventually each person is assigned to an object and the bidding stops. Unfortunately, we do not have the space to go into a detailed description of this algorithm here and must refer the reader to [1]–[3] for a more complete treatment. We will discuss our modifications to this algorithm in terms of the the bidding and assignment phases.

We first modify the bidding phase of [1] by using the time varying benefits of (1) to give us the following calculation:

$$b_{ij_i} = p^i_{j_i}(t) + v_i - w_i + \epsilon = a_{ij_i}(t) - w_i + \epsilon \qquad (6)$$

where $A(i)$ is the set of tasks agent i is capable of completing, $p^i_j(t)$ is agent i's perception of the price of task j, v_i and w_i are the values (benefit minus price) of the best task (j_i) and next best task for that agent, and b_{ij_i} is the final bid of agent i for task j_i which includes a mandatory positive bid increment ϵ (where ϵ is a uniform constant for all agents). Our second modification of the bidding phase is to link the agents' motion to their bidding. Specifically, an agent will always track the optimal trajectory to the task associated with their last bid. This kind of motion control is considered "individually optimistic" in that each agent assumes it has the winning bid for its preferred task. In many situations this sort of behavior will result in the agents moving closer to the tasks to which they eventually be assigned during the progress of the auction, thus reducing the loss of benefit the delay in assignment causes.

The assignment phase of the auction algorithm requires that one and only one agent be responsible for handling the bids for each target (i.e., broadcasting information about which agent won the last bid for that task and its new price). This can be accomplished by either giving a single "leader" agent this auctioneer duty for every task or by partitioning these tasks among two or more agents. Actually, since it is only necessary that just one agent have auctioneer duty for a task *at a time*, this duty can be passed from one agent to another so long as it is guaranteed that no two agents will ever have it simultaneously. For instance, the approach we have taken in our simulations is for the agent with auctioneer duty for a task to pass that duty to the agent that won the last bid for that task.

Our last modification of the the distributed auction algorithm is to require partial asynchronicity [2] (as opposed to the totally asynchronous setting of [1]). Since the prices of the tasks are the main piece of information transferred between the agents (and since these values are non-decreasing) this means that there must exist a finite number D such that for any agent i and any task j,

$$p_j(t-D) \leq p^i_j(t) \leq p_j(t) \qquad (7)$$

with the implication that if $p_j(t) = p_j(t-D)$, then each agent is guaranteed to have perfect knowledge of task j's price.

III. Results

Algorithm Termination and Assignment Optimality

In this section, we will first show that the modified auction maintains an arbitrary ϵ–complimentary slackness (ϵ–CS) condition and then prove that the algorithm terminates in finite time when the bidding increment meets the stated criteria. All the proofs presented in this section are based on those found in [1] and [3] but have been augmented to handle the time-varying benefit function in (1).

Lemma 1: For all $t \geq 0$, the motion control algorithm maintains an ϵ–CS condition for the *partial assignment* S (i.e., a collection of pairs (i,j) where agent i has the highest accepted bid for task j).

Proof: If S satisfies ϵ–CS at time t, then by definition (see [3]) we have

$$a_{ij}(t) - p_j(t) \geq \max_{k\in A(i)} \{a_{ik}(t) - p_k(t)\} - \epsilon\ \ \forall\,(i,j)\in S \qquad (8)$$

so for any Δt such that no reassignments are made between time t and $t+\Delta t$, for every $(i,j)\in S$ we also have

$$\begin{aligned}
&\max_{k\in A(i)} \{a_{ik}(t+\Delta t) - p_k(t+\Delta t)\} - \epsilon \\
&\quad\leq \max_{k\in A(i)} \{a_{ik}(t) - p_k(t)\} - \epsilon \qquad (9)\\
&\quad\leq a_{ij}(t) - p_j(t) \qquad (10)\\
&\quad= a_{ij}(t+\Delta t) - p_j(t+\Delta t) \qquad (11)
\end{aligned}$$

because of inequalities (5), (7), and (8) and since agent i tracks task j. Therefore, ϵ–CS still holds in between reassignments. Now consider a bid made at time t by agent i for task j at a bid price of b_{ij}. Task j must satisfy

$$a_{ij}(t) - p^i_j(t) \geq \max_{k\in A(i)} \{a_{ik}(t) - p^i_k(t))\} - \epsilon\ \ \forall\,(i,j)\in A(i) \quad (12)$$

and b_{ij_i} is calculated to satisfy the above equation if $p^i_j(t) = b_{ij_i}$. If this bid is accepted at time $t+\Delta t$ then because agent i has been tracking task j (and since the auctioneer for task j knows its true price), then the logic behind (9)–(11) still holds and thus ϵ–CS of S is maintained after reassignments as well as in between them and so the lemma holds. □

Theorem 1: When information delays are bounded by a constant D, the motion control algorithm terminates at some $t_a < \infty$ provided there exists at least one feasible full assignment, $\epsilon > 2DB(m-1)(V+1)$, and no vehicle reaches its preferred task before t_a. The final assignment has a collective benefit that is within $m\epsilon$ of the optimal assignment given the configuration of the agents at time t_a.

Proof: We assume the algorithm does not terminate in a finite time and prove the theorem by contradiction. We do this in a manner similar to the termination proofs for the versions of auction algorithm that appear in [3] and [1]. To ensure that enough tasks receive bids to create a full assignment, it was sufficient in [2] and [1] to show that the value of each agent's preferred task eventually fell below the benefit of some unassigned task. We must, however, show that the value of the agent's preferred task falls fast enough in order to overcome the effects of potentially decreasing benefits.

If the algorithm never terminates at least one agent is unassigned at all times and thus some agents submit (and some tasks receive) an infinite number of bids because an unassigned agent is guaranteed to to receive rejection of its last bid within $2D$ time units. Let I^∞ denote the subset of agents that bid indefinitely and let J^∞ denote the subset of tasks that receive an infinite number of bids. There must exist a time index t_∞ such that for all $t > t_\infty$ bidding is confined to I^∞ and J^∞.

For any agent $i \in I^\infty$, if no task in $A(i)$ gets reassigned for D time units, then agent i will know the true price for each task in that set. Thus it will make a valid bid for one of those tasks (i.e., a bid that is at least ϵ greater the task's current price) and this bid must be accepted with another D time units. Thus at least one task from $A(i)$ must receive a successful bid every $2D$ time units. Given that, examine the maximum value agent i associates with tasks in $A(i) \bigcap J^\infty$

$$v_i(t) = \max_{j \in A(i) \bigcap J^\infty} \{a_{ij}(t) - p_j(t)\} \tag{13}$$

and note that after t_∞, the price of the task that achieves this value decreases by no less than the bidding increment ϵ every $2D$ time units. We know that $a_{ij}(t)$ cannot increase and prices do not decrease, so v_i either decreases by at least ϵ every $2D$ time units or by some lesser amount if the value of the second best task is within ϵ of the value of the best task. The latter can happen at most $N \leq m-1$ times (the number of tasks in $A(i) \bigcap J^\infty$) before the quantity $a_{ij}(t) - p_j(t)$ for each task j must be at least ϵ less than the original value of of $v_i(t)$, so

$$v_i(t) \leq \max_{j \in A(i) \bigcap J^\infty} \{a_{ij}(t_\infty) - p_j(t_\infty)\} - \epsilon \left\lfloor \frac{t - t_\infty}{2DN} \right\rfloor$$

where $\lfloor \cdot \rfloor$ is the largest integer smaller than its argument. We proceed by letting $\epsilon = 2DB(m-1)(V+1)\delta$ where $\delta > 1$

$$\begin{aligned} &v_i(t) \\ &\leq \max_{j \in A(i) \bigcap J^\infty} \{a_{ij}(t_\infty) - p_j(t_\infty)\} + \epsilon - \frac{\epsilon}{2DN}(t - t_\infty) \\ &\leq \max_{j \in A(i) \bigcap J^\infty} \{a_{ij}(t_\infty) - p_j(t_\infty)\} + 2DB(m-1)(V+1)\delta \\ &\quad - \frac{2DB(m-1)(V+1)\delta}{2D(m-1)}(t - t_\infty) \\ &\leq -\delta B(V+1)(t - t_\infty) + \max_{j \in A(i) \bigcap J^\infty} \{a_{ij}(t_\infty) - p_j(t_\infty)\} \\ &\quad + 2DB(m-1)(V+1)\delta \end{aligned} \tag{14}$$

so $v_i(t)$ bounded from above by a decreasing affine function of time after t_∞. Since the prices of tasks $j \notin J^\infty$ are constant after t_∞, the value $\overline{v}_i$ agent i associates with the best of those tasks is bounded by

$$\begin{aligned} \overline{v}_i(t) &= \max_{j \notin J^\infty} \{a_{ij}(t) - p_j^i(t)\} \\ &\geq \max_j \{a_{ij}(t_\infty) - B(W + (V+1)(t - t_\infty)) - p_j(t_\infty)\} \\ &= -B(V+1)(t - t_\infty) \\ &\quad + \left[\max_j \{a_{ij}(t_\infty) - p_j(t_\infty)\} - BW\right] \end{aligned} \tag{15}$$

which is also a decreasing affine function of time after t_∞. The rate of decrease in (14) is less than that in (15), so some task $j \notin J^\infty$ must eventually become the best value for each agent $i \in I^\infty$ at some time $t > t_\infty$. Since these tasks never receive a bid after t_∞ it must be the case that none of these tasks are in $A(i)$ for any agent in I^∞, implying that $A(i) \subset J^\infty \ \forall\, i \in I^\infty$ and $\bigcup_{i \in I^\infty} A(i) = J^\infty$. After a finite length of time, every task in J^∞ will be assigned to some agent from I^∞. Since there will still be some agent from I^∞ bidding, there must be more agents in I^∞ than tasks in J^∞, contradicting the assumption that a feasible solution exists. The sub-optimality bound of the final assignment follows from Lemma 1 and a theorem from [3]. □

Because the assumption that no agent reaches a task before the algorithm terminates is difficult to guarantee, and because the worst case analysis of Theorem 1 requires a very large value of ϵ (i.e., much larger than is needed in practicality), we also include the termination results for more relaxed conditions. We first consider the case where the agents will always stay within a certain distance of each task, i.e.,

$$0 \leq |\sigma[s_i(t), d_j]| \leq X \qquad \forall t > 0,\ \forall i, j \tag{16}$$

for some $X > 0$. Inequality (16) implies

$$C_{ij} - BX - Bt \leq a_{ij}(t) \leq C_{ij} - Bt \qquad \forall t > 0\ \forall i, j. \tag{17}$$

Theorem 2: When (17) holds for all $t \geq 0$, the motion control algorithm terminates for any $\epsilon > 0$ provided there exists a feasible full assignment.

Proof: Without loss of generality, assume no tasks are completed before a full assignment is reached. If they are handled correctly (see the next theorem), then an early task completion simply changes the assignment problem to another with one less agent and task. Now for some agent i, the value $v_i(t)$ it associates with its preferred task from the current partial assignment $S(t)$ can be bounded from above by

$$v_i(t) = \max_{j \in S(t)} \{a_{ij}(t) - p_j(t)\} \leq -Bt - \min_{j \in S(t)} p_j(t) + \max_j C_{ij}$$

whereas the value $\overline{v}_i(t)$ that agent i associates with an arbitrary task not in $S(t)$ can be bounded from below by

$$\overline{v}_i(t) \geq -Bt + \min_j C_{ij} - BX.$$

Since the price term $\min_{j \in S(t)} p_j(t) \to \infty$ as $t \to \infty$ (as established in Theorem 1), so long as a full assignment is not reached, $v_i(t)$ will eventually fall below $\overline{v}_i(t)$ and some task will be added to the assignment. This process must repeat until the assignment is full, causing the algorithm to terminate in finite time. □

Theorem 3: Consider the configuration of agents and tasks at a specific instant in time. If: 1) The vectors describing the the agents and the vectors describing the tasks all belong to the same state space, 2) the optimal trajectory distance function satisfies the triangle inequality and $|\sigma[x, x]| = 0$, and 3) there exists an agent $i^\star$ and a task $j^\star$ that are collocated (i.e., $s_{i^\star} = d_{j^\star}$), then the minimum possible total travel time of a one-to-one assignment between agents and tasks is achieved by an assignment containing the pair $(i^\star, j^\star)$. That is to say that letting agents complete tasks prior to algorithm termination under these conditions cannot decrease the collective benefit when $C_{ij} = 0$ for all agent-task pairs.

Proof: Let A be the minimum possible total travel time for a configuration of agents and tasks where there exists agent $i^\star$ and task $j^\star$ such that $s_{i^\star} = d_{j^\star}$. Let S be an assignment that achieves that minimum. Likewise, let $A^\star$ be the minimum possible total travel time for the same configuration and $S^\star$ an assignment that achieves it under the constraint that $(i^\star, j^\star) \in S^\star$. Since S is a less restricted assignment than $S^\star$, it follows that $A \le A^\star$.

Assume $(i^\star, j^\star) \notin S$ (the theorem is trivial otherwise). Let i' be the agent assigned to $j^\star$ and j' the task assigned to $i^\star$ under S. For convenience, let $S' = \{(i^\star, j'), (i', j^\star)\}$. Analyzing the value of A we get

$$\begin{aligned} A &= \sum_{(i,j)\in S} |\sigma[s_i, d_j]| \\ &= |\sigma[s_{i'}, d_{j^\star}]| + |\sigma[s_{i^\star}, d_{j'}]| + \sum_{(i,j)\in S-S'} |\sigma[s_i, d_j]| \\ &= |\sigma[s_{i'}, d_{j^\star}]| + |\sigma[d_{j^\star}, d_{j'}]| + \sum_{(i,j)\in S-S'} |\sigma[s_i, d_j]| \\ &\ge |\sigma[s_{i'}, d_{j'}]| + \sum_{(i,j)\in S-S'} |\sigma[s_i, d_j]| \\ &= |\sigma[s_{i^\star}, d_{j^\star}]| + |\sigma[s_{i'}, d_{j'}]| + \sum_{(i,j)\in S-S'} |\sigma[s_i, d_j]| \\ &\ge A^\star. \end{aligned}$$

Thus $A \le A^\star \le A$ and this implies that $A^\star = A$, proving that the constrained assignment achieves the optimal value. □

Note that when C_{ij} varies widely among agent-task pairs, the early completion of a tasks has the potential to be much more detrimental to the maximum possible benefit. Also, if agents are allowed to complete tasks early, it is important that no other agents waste their resources on the same task and so some mechanism must be put into place to prevent this.

A. Optimality Bounds

In this section, we derive worst case bounds for the collective benefit received from the assignment reached at time t_a for both the motion control algorithm and the original auction algorithm (where benefits are fixed at their initial values and agents wait for the final assignment before tracking their targets). Due to space constraints, we must omit some of the steps in these calculations (contact the authors for details). The proof of Theorem 1 gives us the tools we need to calculate t_a for the motion control algorithm, with the final result

$$t_a \le 2D(m-1)\frac{\max a_{ij}(0) - \min a_{ij}(0) + BW + \epsilon}{\epsilon - 2DB(m-1)(V+1)}. \tag{18}$$

For the original auction algorithm we can adapt the results of [7] and [8] to get

$$t_a \le 2D(m-1)\left(\frac{\max a_{ij}(0) - \min a_{ij}(0)}{\epsilon} + 1\right). \tag{19}$$

For the original auction algorithm a lower bound on the collective benefit of the final assignment follows directly from (19) and $\epsilon - CS$ theorem of [3] as

$$\sum_{i=1}^{m} a_{ij_i}(t_a) \ge A^* - m\epsilon - mB\,(t_a + Y). \tag{20}$$

Where A^* is the maximum achievable benefit and $Y \ge 0$ is the maximum amount of time that can be added to an agents travel time to a task (i.e., $Y > 0$ if the agents cannot remain stationary while waiting for the algorithm to terminate). For the motion control algorithm we use a different methodology to generate a lower bound on the collective benefit that is better than (20) for large delay values.

Let $i^\star, j^\star$ be the last agent-task pair assigned by the motion control algorithm. Since no agent bid on $j^\star$ prior to t_a, the best lower bound on $a_{i^\star j^\star}(t_a)$ we have is $a_{i^\star j^\star}(0) - BW - B(V+1)t_a$. We note two facts concerning the prices at t_a. First, the price of $j^\star$ is still equal to zero. Second, since we are guaranteed to have a price increase of at least ϵ every $2D$ time units, the sum of the prices of the other assigned tasks must satisfy $\sum_{i=1, i\ne i^\star}^{m} p_{j_i}(t_a) \ge \epsilon \lceil t_a/2D \rceil \ge \epsilon t_a/2D$ (where j_i is the final task assignment of agent i and $\lceil\cdot\rceil$ is the smallest integer greater than its argument).

Now consider the benefit terms $a_{ij_i}(t_a)$. Since ϵ-CS is maintained, we know that for each agent i, the following holds:

$$\begin{aligned} a_{ij_i}(t_a) - p_{j_i}(t_a) &\ge \max_k \{a_{ik}(t_a) - p_k(t_a)\} - \epsilon \\ &\ge a_{ij^\star}(t_a) - \epsilon \\ &\ge a_{ij^\star}(0) - BW - B(V+1)t_a - \epsilon. \end{aligned} \tag{21}$$

The collective benefit from agents other than $i^\star$ is

$$\sum_{i=1, i\ne i^\star}^{m} a_{ij_i}(t_a) = \sum_{i=1, i\ne i^\star}^{m} (a_{ij_i}(t_a) - p_{j_i}(t_a)) + \sum_{i=1, i\ne i^\star}^{m} p_{j_i}(t_a). \tag{22}$$

Combining (22), (21), the bound on $a_{i^\star j^\star (t_a)}$, and our observations about prices (and letting $\underline{a} = \min_{i,j} a_{ij}(0)$ and $\overline{a} = \max_{i,j} a_{ij}(0)$) after some rearrangement and bounding, we get

$$\begin{aligned} \sum_{i=1}^{m} a_{ij_i}(t_a) &\ge A^* - m(\overline{a} - \underline{a}) - (m-1)\epsilon - mBW \\ &\quad - B(V+1)t_a - B(V+1)(m-1)t_a + \epsilon\frac{t_a}{2D} \qquad (23) \\ &= A^* - m(\overline{a} - \underline{a}) - (m-1)\epsilon - mBW \\ &\quad - [1 - (\delta - 1)(m-1)]B(V+1)t_a \qquad (24) \end{aligned}$$

where (24) is (23) with $\epsilon = \delta 2DB(V+1)(m-1)$. If δ is at least $1 + 1/(m-1)$ (which for typical m is only slightly larger than strict lower bound of 1 necessary to meet the conditions of Theorem 1) then the collective benefit of the motion control algorithm is bounded from below by the first four terms of (24) *which do not depend on the termination time*. With this value of δ, numerical evaluation shows that (20) decreases at a faster rate with respect to the delay value D than does (24) (although the magnitude of this effect depends significantly on the other system parameters). Thus, the worst-case performace of the motion control algorithm will be better than that of the original auction algorithm for large enough information delays.

IV. Simulations

In this section, we present results from some simulations comparing the motion control algorithm to a centralized assignment algorithm. Omitted from this note (but available upon request from the authors) are several other interesting simulation results. Those other simulations demonstrate the superiority of the motion control algorithm over the distributed auction as the magnitude of delays are increased and also emphasize the benefit of letting agents complete tasks early in total-travel-time minimization problems. In addition, simulations demonstrate the importance of optimizing the value of the bidding increment. In other words, for a specific problem scenario and set of system parameters the expected collective benefit is maximized when ϵ is not too small (which results in a large termination time but does provide optimization with respect to the agents positions at that time) nor too big (which causes the algorithm to terminate quickly but at a poor solution).

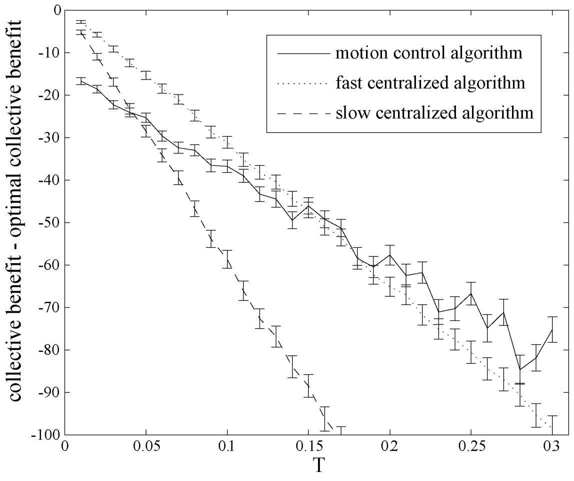

Fig. 1. Comparison of algorithms in restricted bandwidth scenario (10 agents/10 tasks). Error bars are $\pm 3\sigma$ of estimator. 2148 trials per data point.

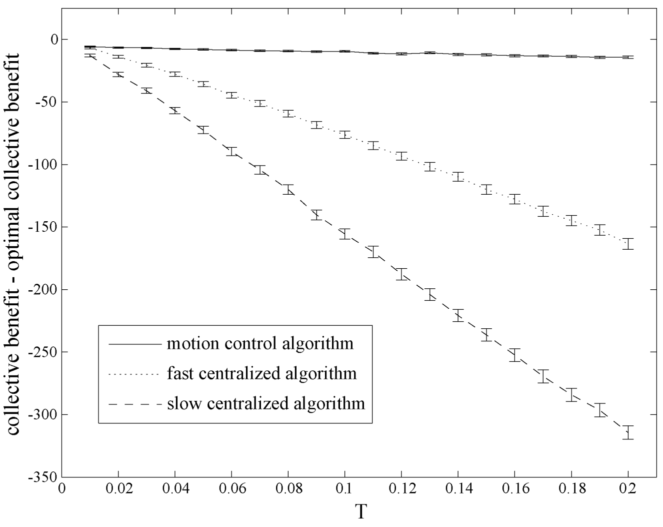

Fig. 2. Comparison of algorithms in restricted bandwidth scenario (10 agents/20 tasks). Error bars are $\pm 3\sigma$ of estimator. 2000 trials per data point.

When information delays are caused primarily by transmission delays (e.g., the physical delay of radio waves or the time required to create and process message headers) then it is probably more efficient to solve our agent-task assignment problem centrally (by having a single leader agent collect all the agents' benefit information, solve the assignment problem by whatever algorithm desired, and then communicate the final assignments) rather than in a distributed fashion (where many messages must be sent back and forth and the transmission delay is incurred many times). On the other hand, if delays are due to restricted bandwidth (i.e., where the amount of information is relatively large compared to the rate at which it can be sent), then a distributed algorithm can sometimes require less information to be exchanged than a centralized one, and hence it may reach an assignment quicker. We illustrate this concept by including the results of a simulation in which ten autonomous air vehicles (modeled as Dubins cars with a speed and turning radius of 100 m/s and 1000 m respectively) wish to minimize the total travel time it takes to get to the tasks (i.e., $C_{ij} = 0$ for all i and j). Agents and tasks were given random initial positions in a 20-km^2 environment, which results in an assignment problem that can usually be solved in a few iterations of the auction algorithm. We compared the motion control algorithm to a centralized algorithm under two assumptions about the amount of data the lead agent would need to collect (we will call the centralized algorithm "fast" when less data is required and "slow" otherwise). For one scenario with 10 available tasks and another with 20, we varied the time in seconds T it took to transmit the amount of data contained in one auction message (where the centralized algorithm took either $(1/4)Tm(2+n)$ or $(1/4)Tm(2+2n)$ seconds to terminate). Fig. 1 displays the results for 10 tasks and clearly shows that as T increases the performance of the motion control surpasses that of the centralized algorithm. This performance gap is even more obvious in Fig. 2 when 20 tasks are available. This is because the additional tasks do not in general cause the motion control algorithm to take longer (in this scenario) but do dramatically increase the amount of information that must be collected to solve the problem centrally.

V. Conclusion

In this work, we have sought to address an assignment problem between mobile agents and stationary tasks where the benefits (and hence the optimal assignment quality) have the potential to decrease during the time used to calculate a solution. We presented a modification of the distributed auction algorithm of [1] that controls the motion of the agents during the algorithm's progress in an attempt to minimize that loss of benefit. We showed that this algorithm is guaranteed to terminate in finite time at an assignment that is within a known bound of the optimal solution under one set of assumptions, and simply guaranteed to terminate under less restrictive conditions. Simulations have demonstrated that our modified algorithm can be superior to a centralized approach in situations where communication is hampered by bandwidth restrictions.

References

[1] D. P. Bertsekas and D. A. Castañon, "Parallel synchronous and asynchronous implementations of the auction algorithm," *Parallel Comput.*, vol. 17, pp. 707–732, 1991.
[2] D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Belmont, MA: Athena Scientific, 1997.
[3] D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Belmont, MA: Athena Scientific, 1998.
[4] D. A. Castañon and C. Wu, "Distributed algorithms for dynamic reassigment," in *Proc. 42nd IEEE Conf. Decision Control*, Maui, HI, Dec. 2003, pp. 13–18.
[5] C. Schumacher, P. R. Chandler, and S. J. Rasmussen, "Task allocation for wide area search munitions via iterative network flow," in *Proc. AIAA Guid., Navigation, Control Conf.*, Monterey, CA, 2002, 2002–4586.
[6] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal position," *Amer. J. Math*, vol. 79, pp. 497–516, 1957.
[7] D. P. Bertsekas, Auction algorithms for network flow problems: A tutorial introduction Lab. Inform. Decision Syst. Mass. Inst. Technol., Cambridge, MA, Tech. Rep. LIDS-P-2108.
[8] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA: Athena Scientific, 1997.