Coping with Information Delays in the Assignment of Mobile Agents to Stationary Tasks

Brandon J. Moore *Member, IEEE* and Kevin M. Passino *Fellow, IEEE*

Abstract

Cooperative control involves the development of decision-making systems that can guide the behavior of a group of networked autonomous agents toward a common goal. This problem is often put in the context of assignment of agents to tasks, and these tasks are often spatially distributed in an environment through which the agents must navigate. In this paper we consider the problem of determining a one-to-one assignment of mobile agents to stationary tasks in order to maximize a performance measure that is sensitive to the time each task will be completed. The problem is complicated by the fact that the agents must communicate over a network with substantial delays in order to cooperatively determine their assignment. Thus the problem is similar to the traditional assignment problem, with the exception that the benefit values can change while the agents attempt to determine a solution. A modification to the asynchronous distributed auction algorithm is developed to solve this problem and shown to terminate in finite time. Optimality of the resulting assignment is investigated through a worst case analysis and via Monte Carlo simulations using agents based on the Dubins car model.

I. Introduction

Within the field of cooperative control there is special interest in problems where a group of mobile agents must act in concert to accomplish a common goal. Generating the individual agent trajectories and associated actions that accomplish this goal can be viewed as the assignment of each agent to a certain subset of spatially distributed tasks. Since the agents will generally not have the same initial position or capabilities, it follows that some assignments will result in better performance than others and thus we would like to determine the optimal one subject to the problem's constraints. Given the mobile nature of the agents, the performance measure used is often some function of the distance each agent travels or the times that various tasks are completed. In this work, we focus on a distributed method autonomous agents may use in order to solve a particular type of assignment problem despite issues that arise when inter-agent communications are subject to significant delays.

This work was supported by the AFRL/VA and AFOSR Collaborative Center of Control Science (Grant F33615-01-2-3154). In addition, the work was supported by a DAGSI Fellowship to B. Moore.

B. Moore is with the Department of Electrical and Computer Engineering, The Ohio State University (e-mail: mooreb@ece.osu.edu)

K. Passino is with the Department of Electrical and Computer Engineering, The Ohio State University (e-mail: passino@ece.osu.edu)

To date, research on these types of problems has focused on either optimization algorithms or stability analysis. In the former case, the application of mixed-integer linear programming (MILP) has proved very useful when applied to the optimization of agent trajectories [1] or to coordinating the efficient interaction of multiple agents in scenarios with many sequential tasks and tight timing constraints [2], [3]. Given the potential complexity of these problems, however, the solution time of these algorithms can become prohibitive. In some cases the optimality of the solution may degrade gracefully with the time of its computation. In other cases, particularly those involving the popular nonholonomic model known as the Dubins Car [4], the feasibility of the solution may be destroyed if it takes too long to compute it. By their nature, such algorithms are also highly centralized; this often means that a large amount of information must be passed over the agents' communication network (a potential problem if that network is subject to imperfections such as delays, noise, or changing topology). Some of these issues can be overcome by settling for suboptimal solutions either through the use of heuristics (such as those developed in [5]) or by decomposing the larger problem into smaller ones that can each be solved optimally and then recombined to generate a complete solution (as in [6]). Through the use of Monte Carlo simulations, statistics can be generated to check if these algorithms are "good enough," but this may fail to identify pathological situations in which they perform poorly or even fail, particularly when unrealistic assumptions are made concerning communications.

When we start to consider communication imperfections, it often makes sense to forgo optimality of the assignment in favor of stability. That is, given asynchronous communications with delays and the resulting information imbalances between agents, it is no longer trivial to assume an algorithm will accomplish its goal. Stability can be defined in many ways. In the next paragraph we discuss this concept in terms of termination (i.e. guarantees that an algorithm will eventually produce a feasible solution), but there are equally important definitions more along the lines of traditional control theory. In [7], a group of agents must continually processes a set of reoccurring tasks, so in this context it makes sense to show that their control law guarantees an upper bound on the longest time any particular task will be ignored. In [8], a load balancing approach is used to divide a number of tasks between agents. Here they prove that an equilibrium set (i.e. all agents balanced within a certain range) is exponentially stable in the large. In this scenario, if the agents were capable of balancing perfectly, then this approach would also result in an optimal solution; however, the discrete nature of the load means it can only be balanced to within a fairly wide margin of error. In our work, we have attempted to find a middle ground between the two approaches of optimality and stability; the algorithm we present cannot guarantee optimality due to the unavoidable effects of communication delays, but does seek to minimize the harm done to performance and does so in a way that allows us to analytically quantify a guaranteed performance level.

The problem we focus on in this paper is a variant of the assignment problem from the field of combinatorial optimization. This problem formulation has received much attention in cooperative control studies because it models the need to generate a suitable mapping between two finite sets subject to certain constraints, and many algorithms of polynomial complexity exist to solve it [9], [10], [11]. In [12], the distributed sequential shortest augmenting path is modified in a manner that allows the agents to cooperatively compute an optimal solution despite information delays between agents and the dynamic arrival of additional tasks. The existing algorithms for the assignment

problem have also been used heuristically to deal with tasks that are highly coupled (i.e., the completion of certain tasks must be preceded by the completion of another). In [6], multiple assignment problems with evolving subsets of the agents and tasks are iteratively solved, thus allowing their algorithm to produce a multiple task tour solution that simulations show to be close to optimal for their scenario. That work uses a centralized redundant approach in which each agent solves the same problem independently and then implements its own portion of the resulting assignment. It is therefore highly dependent on the agents starting from the same information in order to ensure they all arrive at the same solution. This becomes problematic when communication delays are introduced; because of changes in the problem that may occur during planning (i.e., vehicle movement), the final solution may no longer be optimal or even feasible. In this work we seek to incorporate these changes in problem data directly into a modification of the distributed auction algorithm of [9] for determining an optimal solution to the standard assignment problem. To the best of our knowledge, this is the first time an assignment method has explicitly dealt with a performance measure of a time-varying nature.

This paper is organized as follows. In Section II we present some background on the assignment problem and the asynchronous distributed auction algorithm of [9] in order to establish some notation for the paper. In Section III we define the specific problem to be solved and in Section IV develop a modification of the distributed auction to solve it. We determine the conditions under which that algorithm is guaranteed to terminate and perform a worst case analysis of the resulting solution. Section V contains simulation results demonstrating the practical performance of the modified algorithm. Conclusions appear in Section VI.

II. Background

The standard *assignment problem* is one in which members of one set must be matched to members from another distinct set on a one-to-one basis. By convention [11], the former set is usually referred to as the set of *persons* and the latter as the set of *objects*. An *assignment* is a collection of pairings such that each person and each object appears in at most one pair. An assignment is considered *full* if it contains every member from the smaller of the two sets (persons or objects) and *partial* otherwise. For each possible pairing between a person and an object, there exits some scalar value that represents the *benefit* gained if that pair is a member of an assignment. The sum of the benefits associated with each pair in an assignment is referred to as its *collective benefit*.

The goal of an assignment problem is to find a full assignment that is optimal in that it maximizes the collective benefit of the individual pairings (i.e., their sum). If the sizes of the person and object sets are the same (making it so that every person and object must participate in any full assignment) we have a special case referred to as the *symmetric assignment problem*. In this paper, we consider an *asymmetric assignment problem* where the number of objects is greater than or equal to the number of persons, thus allowing for the possibility that some objects may be excluded from a full assignment. We give a brief overview of the assignment problem in this section; for a more thorough discussion see [11] or [10].

A. Primal and Dual Problems

Let $\mathcal{A}$ be the collection of all pairs (i, j) denoting an allowable assignment between person i and object j and let a_{ij} be the scalar benefit associated with that pair. If object j is assigned to person i, then let a decision variable x_{ij} be equal to 1 (and equal to 0 otherwise). Since we wish to consider an asymmetric assignment with more objects than persons, we introduce another decision variable x_{sj} which is equal to 1 if object j is not assigned to any person (in which case we say it is assigned to a *supersource* instead). With this notation, the asymmetric assignment problem between m persons and n objects can be stated as the constrained optimization problem,

$$\text{maximize} \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \tag{1}$$

subject to

$$\sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} = 1, \forall i \in \{1,\dots,m\} \tag{2}$$

$$\sum_{\{i|(i,j)\in\mathcal{A}\}} x_{ij} + x_{sj} = 1, \forall j \in \{1,\dots,n\} \tag{3}$$

$$\sum_{j=1}^{n} x_{sj} = n - m, \tag{4}$$

$$x_{ij}, x_{sj} \in \{0, 1\} \tag{5}$$

Where (1) is the collective benefit and the constraints in (2)-(5) ensure that each person is assigned to one unique object. This formulation is known as the *primal problem*. Due to the linear nature of the problem, it is possible to form the *dual problem* given by (see [11]),

$$\text{minimize} \sum_{i=1}^{m} \pi_i + \sum_{j=1}^{n} p_j - (n-m)\lambda$$

$$\text{subject to} \quad \begin{array}{l} \pi_i + p_j \geq a_{ij}, \quad \forall (i,j) \in \mathcal{A} \\ \lambda \leq p_j, \quad \forall j = \{1,\dots,n\} \end{array}$$

where π_i, p_j, and λ, are Lagrange multipliers associated with each person i, each object j, and the supersource, respectively. For convenience, let $\pi = [\pi_1, \dots, \pi_m]$ and $p = [p_1, \dots, p_n]$. The structure of the two problems dictates that the decision variables x_{ij} are completely dependent upon π, p, and λ. Therefore, once we have a solution to the dual problem, we can easily reconstruct the solution to the primal problem as

$$x_{ij} = \arg \min_{w_{ij}\in\{0,1\}} (a_{ij} - \pi_i - p_j) w_{ij}$$

The above result and indeed the motivation for forming the dual problem comes from existence of the *complimentary slackness* (CS) criteria [11]. This criteria allows us to determine whether a given assignment is the optimal solution, and provides a guide for constructing such an assignment. Specifically, we use a relaxed criteria, $\epsilon-$*complementary slackness* ($\epsilon-$CS), that allows us to generate assignments that are suboptimal but whose collective benefit is within a proscribed bound of the optimal solution. An assignment S is said to satisfy $\epsilon-$CS for the asymmetric assignment

problem if

$$\pi_i + p_j \geq a_{ij} - \epsilon, \quad \forall (i,j) \in \mathcal{A} \tag{6}$$

$$\pi_i + p_j = a_{ij}, \quad \forall (i,j) \in S \tag{7}$$

$$p_j \leq \min_{k \text{ assigned under } S} p_k, \quad \forall j \text{ unassigned under } S \tag{8}$$

where (6) and (7) are often combined to yield

$$a_{ij} - p_j \geq \max_{k \in A(i)} \{a_{ik} - p_k\} - \epsilon \quad \forall\ (i,j) \in S \tag{9}$$

with $A(i) = \{j | (i,j) \in \mathcal{A}\}$.

Theorem 1: [11] If a feasible full assignment S along with a specific π and p satisfies $\epsilon-$CS, then the collective benefit of S is within $m\epsilon$ of the optimal solution to the asymmetric assignment problem.

B. Auction Algorithm

The auction algorithm [11], [10] is a method of solving the assignment problem using elements from both the primal and dual formulations that is motivated by an economic interpretation of the $\epsilon-$CS condition. The variables p and π are thought of as the *prices* and *profits* of the objects and persons, respectively. The $\epsilon-$CS conditions are then interpreted as stating that if person i is assigned to object j, then his profit is equal to the benefit he receives from that object minus its price (7), and that object j provides person i more profit (within some tolerance ϵ) than any other object (9).

The auction algorithm starts with an empty assignment S (which trivially satisfies $\epsilon-$CS) and iteratively modifies it by either adding new assignment pairs or replacing current ones while adjusting prices and profits in a manner that maintains $\epsilon-$CS. Thus, the size of S is strictly non-decreasing as the algorithm progresses, and when a full assignment is reached it has the optimality guarantees of Theorem 1. The modification of the working assignment takes place by having unassigned persons bid against each other for objects (or objects for persons), thereby raising their prices (profits) much in the same way as a real auction. Bidding decisions are based on *values*, which are the differences between the benefits a bidder associates with objects (persons) and their prices (profits).

Although many variations of the auction algorithm exist, this paper will be concerned exclusively with the forward auction algorithm in which persons bid for objects and prices are the only variable of concern (π and λ are not directly altered and can easily be calculated from other information). Under mild restrictions this algorithm can be used even in a distributed asynchronous manner to solve the asymmetric assignment problem in a finite period of time [9]. The algorithm consists of two main operations, *bidding* and *assignment*, which are described below.

Bidding: Let U be the (nonempty) set of persons unassigned under a partial assignment S and $R \subseteq U$ be those persons with a pending bid. Let p_j^i represent person i's most recent knowledge about the true price of object j, p_j. To submit a bid, person $i \in U - R$ takes the following steps:

1) Finds its preferred object j_i (the one that returns the best possible value v_i)

$$j_i = \arg \max_{j \in A(i)} \{a_{ij} - p_j^i\}$$

$$v_i = \max_{j \in A(i)} \{a_{ij} - p_j^i\}$$

as well as the value of the next best object w_i

$$w_i = \max_{\substack{k \in A(i) \\ k \neq j_i}} \{a_{ik} - p_k^i\}$$

(if j_i is the only object in $A(i)$, let $w_i = -\infty$).

2) Calculates its bid for object j_i as

$$b_{ij_i} = p_{j_i}^i + v_i - w_i + \epsilon = a_{ij_i} - w_i + \epsilon$$

and communicates this information to object j_i.

Assignment: Let $P(j)$ be the set of persons with bids pending for object j since the last time it was reassigned (if ever). If $P(j)$ is not empty, then object j takes the following steps to resolve the bidding:

1) Selects the person i_j from $P(j)$ with the highest bid $b_{i_j j}$

$$i_j = \arg \max_{i \in P(j)} b_{ij}$$

$$b_{i_j j} = \max_{i \in P(j)} b_{ij}$$

2) If $b_{i_j j} \geq p_j + \epsilon$ then it sets $p_j := b_{i_j j}$, updates the partial assignment S by replacing the current person assigned to it (if any) with person, i_j, and communicates this new information to all persons.
3) If $b_{i_j j} < p_j + \epsilon$ then all bids for object j are cleared, no reassignment or price change is made, and it communicates this information to all persons in $P(j)$.

From the description above, we can see that prices are strictly increasing and that once an object becomes assigned, it remains assigned for the duration of the algorithm. Thus, if the initial prices of the objects are identically zero, the $\epsilon-$CS condition of (8) will be automatically satisfied because the price of any assigned object must be at least ϵ.

It should be noted that "person i bids…" and "object j assigns…" each refer to calculations done by a specific processor. Which processor is responsible for performing the calculations for a specific person or object at a given moment in time will depend on what makes the most sense considering the application. While there is a great deal of flexibility, the need to ensure the integrity of the algorithm places some limitations on the methods we may use. It is imperative that the previously mentioned restrictions are met, and, in a distributed processing environment, the reliability of the communication network and of the processors themselves may be of concern. Therefore, there must be protocols in place that ensure price information is (eventually) transmitted correctly to every processor. Also, if one processor fails, then there must be some form of redundancy built into the system to ensure that another processor will assume its responsibilities. However, adding such redundancy can create problems of its own. If two processors with different knowledge about the current price of an object simultaneously make two different assignments for that object, then there will be conflicting information in the system that must be resolved somehow. Clearly, it is preferable to avoid this hazard by implementing a method ensuring that one and only one processor

may take action for a particular person or object at a time. That processor must also be guaranteed to have correct information concerning that person or object (i.e., a person's assignment or bidding status and an object's true price).

III. Problem Statement

In this work, we will consider a very specific case of the general task assignment problem in the context of cooperative control. We assume that there are m mobile agents of possibly different type that are initially distributed in the environment. There are n tasks ($n \geq m$) at fixed locations, also of possibly differing type. An agent is considered to have completed a task when it arrives there, where arrival can consist of more or less restrictive conditions than mere collocation (e.g., a particular orientation or earliest arrival time are common specifications). Agents are non-renewable resources in that they can perform one and only one task (e.g., a "kamikaze" attack by an autonomous munition). Whether or not an agent has the capability to perform a specific task depends on both its type and the type of that task. Upon completion of a task, each agent receives a benefit that depends on both its type and the target's type and is decremented by a linear function of the time of completion. Optimization for this problem will mean maximizing the collective benefit received by the agents upon completion of m of the tasks, with zero benefit received from the $n - m$ unfinished tasks.

Given the one-to-one nature of the above problem, if the location of initial positions of the agents and tasks are known prior to deployment (i.e., the start of the scenario), then it clearly can be modeled as an asymmetric assignment problem (see Section II) and solved accordingly. Consider, however, the case where the agents must calculate an assignment "on-the-fly" because they lack complete knowledge of the situation prior to deployment. If communications between the agents are subject to substantial delays and/or the agents' available computing power is limited, it could easily take a nontrivial length of time to arrive at an assignment, during which the benefits each agent associates with each task have most likely changed.

A. Feasible and Optimal Vehicle Trajectories

Because the benefits for the assignment problem above are based on the time an agent arrives at a task, in order to understand how those benefits change as the agents move, we must first consider the vehicle dynamics involved. If we define $s_i(t)$ as a state vector containing all the information needed to represent an agent i at a particular point in time (i.e., physical coordinates plus possibly other pertinent data such as orientation), then $s_i(t), t \geq 0$ denotes the state space trajectory an agent follows as a function of time. A trajectory $s_i(t), t \geq 0$ is *feasible* so long as its components satisfy the vehicle dynamics of the agent, which we will usually state in the form of a system of differential equations and associated constraints.

We define the constant d_j as a vector associated with task j and representing the pertinent information describing its location and arrival criteria. If the agents' only concern is to reach the tasks (i.e. collocation) then there is no need for this arrival criteria. In general, however, we allow for its inclusion in order to provide flexibility in modeling more complex situations. For example, an agent might have to arrive at a task along a certain heading.

Alternatively, the agent may be allowed to perform the task from any of a number of different points defined in relation to the nominal location of the 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 (i.e., the set $\mathcal{A}$ of pairs (i,j) denoting agent i is capable of completing task j in the network flow model of Section II is time invariant). 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. Let us denote that trajectory by $\sigma[s_i(t), d_j]$ and its travel time by the metric $|\sigma[s_i(t), d_j]|$. Since the tasks are stationary, by the definition of optimality the travel time of the optimal trajectory to any task from a point $s_i(t+\Delta t)$, can be no less than the difference between the travel time of the optimal trajectory to the same task and the elapsed time interval $\Delta t > 0$. Using our notation,

$$|\sigma[s_i(t+\Delta t), d_j]| \geq |\sigma[s_i(t), d_j]| - \Delta t \qquad \forall\, t \geq 0,\ \ \forall\, 0 \leq \Delta t \leq |\sigma[s_i(t), d_j]| \tag{10}$$

with equality if and only if $\sigma[s_i(t+\Delta t), d_j]$ is a subpath of $\sigma[s_i(t), d_j]$. When that is the case, we will say that agent i is *tracking* task j from t to $t+\Delta t$.

For this work we will require that for any feasible trajectory agent i might follow, the increase in the travel time of the optimal trajectory to a given task j can be upper bounded as

$$|\sigma[s_i(t+\Delta t), d_j]| \leq |\sigma[s_i(t), d_j]| + W + V\Delta t \qquad \forall\, t,\ \ \forall\, \Delta t \geq 0 \tag{11}$$

where the constants $W \geq 0$ and $V \geq -1$ are identical for all i and j. An inequality of this form should be satisfied for many situations with realistic vehicle dynamics (i.e., anywhere the difference $|\sigma[s_i(t+\Delta t), d_j]| - |\sigma[s_i(t), d_j]|$ has a bounded derivative and only limited discontinuities). Consider the following examples:

Pivoting Robot: An agent in a planar environment that can either move forward along its current heading or pivot in place has a kinematic model given by

$$\begin{aligned} \dot{x} &= v\delta[u]\cos\theta \\ \dot{y} &= v\delta[u]\sin\theta \\ \dot{\theta} &= \omega_{max}u \end{aligned} \qquad \text{subject to:} \quad \begin{aligned} 0 &\leq v \leq v_{max} \\ -1 &\leq u \leq 1 \end{aligned}$$

where v_{max} and ω_{max} are the maximum forward and radial velocities respectively, and $\delta[u]$ is the Kronecker delta function ($\delta[u]$ equals 1 if $u = 0$ and 0 otherwise). Optimal trajectories for this type of vehicle consist of a pivot (of up to half a revolution) to face the robot towards its target destination followed by forward travel at its maximum velocity until it gets there. Therefore, the length of the optimal trajectory satisfies an inequality in the form of (11) with $V = 1$ and $W = \frac{\omega_{max}}{\pi}$. □

Dubins Car: Uninhabited air vehicles (UAVs) are often modeled as agents in a planar environment that are forced to move at a constant forward velocity and posses limited steering ability. This can be represented by the kinematic model known as the Dubins Car [4], [13] given by

$$\begin{aligned} \dot{x} &= v_c\cos\theta \\ \dot{y} &= v_c\sin\theta \\ \dot{\theta} &= \omega_{max}u \end{aligned} \qquad \text{subject to:} \quad -1 \leq u \leq 1 \tag{12}$$

where $v_c > 0$ is the fixed forward velocity and ω_{max} is the maximum radial velocity of which the vehicle is capable while traveling at v_c. Optimal trajectories for this type of vehicle consist of movement along paths made up of straight line segments or arcs of radius $R_{min} = \frac{v_c}{\omega_{max}}$. If the arrival criteria for the tasks prescribe a specific orientation, then the length of the optimal trajectory satisfies an inequality in the form of (11) with $V = 1$ and $W = \frac{(2+3\pi)R_{min}}{v_c}$ (see Appendix). While somewhat secondary to our main focus, that derivation demonstrates the applicability of this work to an important class of vehicles and implies that the assumption of (11) is not unreasonable. □

We note that different approaches to determining the pair (V, W) may, in some cases, produce values that are not trivially different (i.e., one approach may produce a smaller V than another, but at the expense of a larger W). In these instances, we have the freedom to choose the pair that gives the best results for the application. For example, if we are concerned with large time intervals the $V\Delta t$ term will dominate (11), and we should select a pair with a smaller V if possible. If, on the other hand, we know the time interval is small, we may want to chose a pair with small W. Of course, we can also combine the inequalities of the different pairs provided the resulting piecewise linear bound is of some use.

B. Benefit Functions

Now that we have a method to characterize how the movement of agents affects their ability to reach the tasks, we propose a motion-dependent formulation of the benefit received by agent i from task j,

$$a_{ij}(t) = C_{ij} - B(t + |\sigma[s_i(t), d_j]|) \tag{13}$$

where C_{ij} is a constant particular to the pairing (i, j) (i.e., a function of the types of both the agent and the task), 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 tracks task j from time t onward. (A more general formulation replaces the constant B with values B_{ij} for each agent/task pair and is considered in Section IV-C). We assume that our algorithm starts at $t = 0$, so this benefit function is equivalent to saying that each task has its nominal benefit C_{ij} discounted proportional to the time it takes for an agent to complete it. If the algorithm starts at some time other than $t = 0$, each benefit function $a_{ij}(t)$ can be scaled accordingly, but this is unnecessary since only the relative benefit between tasks matters when finding the optimal assignment. If we want to take into account that tasks have been waiting to be completed for varying amounts of time before $t = 0$ then we can incorporate relative shifts into the appropriate values of C_{ij}. Note that this seems to be the first time that solving an assignment problem with time-varying benefits has been addressed.

With the benefit function defined above, using the inequalities in (10) and (11) we can place the following bounds on a_{ij},

$$a_{ij}(t + \Delta t) \geq a_{ij}(t) - B(W + (V + 1)\Delta t) \quad \forall \quad t, \Delta t \geq 0 \tag{14}$$

$$a_{ij}(t + \Delta t) \leq a_{ij}(t) \quad \forall \quad t \geq 0, \quad \forall \quad 0 \leq \Delta t \leq |\sigma[s_i(t), d_j]| \tag{15}$$

where equality is reached in (15) if and only if agent i is tracking task j from time t to time $t+\Delta t$. The qualification in (15) 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 stop and wait, 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 remains 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 (15) that the collective benefit of the assignment at t_a is the same as what the agents will receive when they complete their tasks. Equally obvious from these two inequalities is that 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 traditional auction algorithm proposed in the next section attempts to do just this.

IV. Assignment Algorithm and Results

A. Assumptions

In this section we will consider two sources of delay: one which arises from limited computing power and another which comes from the communication network. For computational delays we assume that there exists a maximum bound D_{calc} on the time it takes to complete all the calculations associated with an iteration of the algorithm (i.e., the time from when an agent receives new information until the time it is ready to transmit the results based on that information). If desired, this delay could be described in terms that were proportional to the size of the problem (i.e., the number of tasks). When considering communication delays, the specific network topology and protocols used are unimportant so long as information transmitted by an agent is guaranteed to reach every other agent without error and within a known maximum delay D_{comm}. This delay could also be broken down into the sum of a value representing the maximum transmission delay and a term proportional to the amount of data being sent. Given the low computational complexity of the individual operations used in the auction algorithm and a desire to focus on situations involving significant communication delays, we assume from this point on that $D_{calc} \ll D_{comm}$ and just use $D = D_{calc} + D_{comm}$ as the maximum length of time it takes one agent to receive updated information from another.

We will also assume that every agent knows d_j or an acceptable approximation thereof for all the tasks prior to the start of the algorithm. In addition, it is important that the numbering scheme used to identify the targets has been agreed upon as well, or that there exists a "natural" way of distinguishing targets (e.g., by their coordinates if they lie far enough apart to avoid confusion). While this last assumption is not trivial, it is not the emphasis of this work. Under these assumptions, we need only concern ourselves with information directly related to the operation of the auction algorithm (i.e., bids, prices, and assignment updates). For prices in particular, the maximum

information delay D and the fact that prices never decrease in a forward auction allows us to put the following bounds on agent i's knowledge of the price of task j,

$$p_j^i(t) \leq p_j(t) \tag{16}$$

$$p_j^i(t) \geq p_j(t-D) \tag{17}$$

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.

B. Motion Control for a Distributed Auction

In this section we propose a partially asynchronous algorithm that can be used to solve an assignment problem with motion dependent benefits. This algorithm is a modification of the distributed asynchronous auction presented in [9] that we summarized in Section II:

- Starting with an empty partial assignment S we add or replace agent/task pairs (i, j) based on competing bids from the individual agents that are communicated across the network. This process terminates when every agent has a task assignment.
- Each unassigned agent calculates its own preferred task and transmits its associated bid using a procedure identical to that of Section II but with the exception that they will have to recalculate the benefits $a_{ij}(t)$ each time. We require that each agent perform this process immediately upon learning that it is eligible to bid (i.e., after becoming unassigned, having its last bid rejected, or learning about an assignment update that makes its pending bid obsolete).
- Some agents will be designated to perform the assignment calculations for the tasks, with one and only one agent responsible for each task at a time. We require that each agent performs this task immediately upon receiving a bid, and that it immediately transmits the results (i.e., new price and assignment and/or bid rejection messages).
- The motion of the agents is controlled by two simple rules throughout the duration of the algorithm. If an agent/task pair (i, j) belongs to the partial assignment S, then agent i tracks task j. If an unassigned agent i has a bid pending for task j, then agent i also tracks task j.
- In order to guarantee termination of the algorithm, we will require that the bidding increment used exceed a certain threshold specified in terms of the problem's parameters.

C. Assignment Algorithm Termination

Given this description of the algorithm, we will first show that it maintains an arbitrary $\epsilon-$CS condition for the partial assignment S, 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 [9] and [11] but have been augmented to handle the time-varying benefit function in (13).

Lemma 1: For an assignment S that satisfies $\epsilon-$CS at time t, if every agent in S follows the time optimal trajectory to its assigned target then S will satisfy $\epsilon-$CS at time $t+\Delta t \geq t$ when S is constant from t to $t+\Delta t$, benefits are defined as in (13), and no agent reaches its assignment before $t+\Delta t$.

Proof: If a partial assignment S satisfies $\epsilon-$CS (see Section II) at time t, then for each assignment pair we have

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

Consider the quantity $\max_{k \in A(i)} \{a_{ik}(t+\Delta t) - p_k(t+\Delta t)\} - \epsilon$,

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

where the first line makes use of the the fact that $a_{ij}(t+\Delta t)$ is upper bounded by $a_{ij}(t)$, the second uses the fact that prices are strictly increasing (i.e., $p_k(t) \leq p_k(t+\Delta t)$), the third uses the definition of the $\epsilon-$CS condition, and the last line follows from the stipulation that every agent in S tracks its assigned task from time t to time $t+\Delta t$ and the fact that the price of a task cannot change unless the assignment does. The resulting inequality shows that $\epsilon-$CS still holds at time $t+\Delta t$ for every agent-task pair continuously in the assignment from t to $t+\Delta t$. □

Lemma 2: Providing that an agent tracks the minimum-time path to the task it is currently bidding on, the proposed algorithm maintains $\epsilon-$CS for a partial assignment S for all $t \geq 0$ when benefits are defined as in (13).

Proof: Consider agent i which has just submitted a bid b_{ij_i} for his preferred task j at time t. Since agent i is working with possibly outdated price information, its preferred target satisfies

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

where b_{ij_i} is calculated to satisfy the above equation if $p_j^i(t) = b_{ij_i}$. Since agent i tracks task j while its bid is pending, then by logic analogous to that used in Lemma 1 and the fact that the auctioneer for task j knows its true price, the $\epsilon-$CS condition is guaranteed for pairs entering the partial assignment and thus the overall $\epsilon-$CS of S is maintained. □

Theorem 2: For a feasible asymmetric assignment problem where there exists at least one full assignment S such that every pair (i,j) in S agent i is capable of completing task j, if the benefits $a_{ij}(t)$ are defined as in (13) and information transmission delays are bounded by D, the proposed algorithm terminates in finite time t_a provided that $\epsilon > 2DB(m-1)(V+1)$ and no agent reaches its preferred task before t_a. Furthermore, the final assignment has a corresponding benefit that is within $m\epsilon$ of the optimal assignment given the configuration of the agents at time t_a.

Proof: We first assume that the algorithm does not terminate in a finite amount of time t_a and then prove the theorem by contradiction. We do this in a manner similar to the termination proofs for the standard auction algorithm that appear in [10] and [9]. To ensure that enough tasks receive bids to create a full assignment, it was sufficient

in [10] and [9] 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 does not terminate, it must be the case that at least one (but not necessarily the same) agent is unassigned at all times. Therefore, it is also the case that some agents submit (and thus some tasks receive) an infinite number of bids. This is ensured by the stipulation that unassigned agents must bid immediately if eligible to do so and are guaranteed to receive confirmation or rejection within at most $2D$ time units, after which time they must bid again unless they were accepted into the assignment. If accepted into the assignment, they either displace another agent (who must then bid) or add another agent/task pair to the assignment (which can only happen a finite number of times if the algorithm does not terminate since the size of assignment is strictly non-decreasing).

Let us denote the subset of agents that bid indefinitely as I^∞ and the subset of tasks that receive an infinite number of bids as J^∞. There must then be a sufficiently large time index t_∞ such that for all time $t > t_\infty$ bidding is confined to I^∞ and J^∞.

Consider an agent $i \in I^\infty$ and the set of tasks $A(i)$ to which it may be assigned. Let t_0 be the time a task from $A(i)$ gets reassigned. If no other tasks from $A(i)$ get reassigned before $t_0 + D$, then agent i will have perfect knowledge of the prices for those tasks. With such knowledge, agent i can make a bid for some task in $A(i)$ that will be accepted no later $t_0 + 2D$. Therefore, at least one task from $A(i)$ must receive a successful bid every $2D$ time units.

Given that information, examine the maximum value agent i associates with tasks that are in both $A(i)$ and J^∞,

$$v_i(t) = \max_{j \in A(i) \bigcap J^\infty} \{a_{ij}(t) - p_j(t)\} \tag{18}$$

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. Since the upper bound on the benefits in (15) tells us that $a_{ij}(t)$ cannot increase, and the algorithm dictates that prices do not decrease, this means that 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 we can over bound $v_i(t)$ with

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

where $\lfloor \cdot \rfloor$ is the floor operator (which takes the value of the largest integer less than its operand). Since $-x\lfloor \frac{t}{y} \rfloor$ is a descending staircase function bounded from above by $x - \frac{x}{y}t$, we have

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

Substituting $\epsilon = 2DB(m-1)(V+1)\delta$ where $\delta > 1$ (so that ϵ satisfies the assumed bound) and noting that N can be no greater than $m - 1$ or else at least as many tasks as there are agents would receive an infinite number of

bids (in which case the algorithm would have terminated),

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

Cancelling like terms and collecting the constants on the right gives us

$$\begin{aligned} v_i(t) &\leq -\delta B(V+1)(t - t_\infty) + \max_{j \in A(i) \bigcap J^\infty} \{a_{ij}(t_\infty) - p_j(t_\infty)\} \\ &\quad + 2DB(m-1)(V+1)\delta \end{aligned} \tag{19}$$

which tells us that the value of the best object from the set $A(i) \bigcap J^\infty$ can be bounded by a decreasing affine function of time after t_∞.

Now, since the prices of tasks $j \notin J^\infty$ stop changing after time t_∞, using the lower bound for the change in a benefit from (14), the value $\overline{v}_i$ agent i associates with the best of those tasks is lower bounded by

$$\begin{aligned} \overline{v}_i(t) &= \max_{j \notin J^\infty} \{a_{ij}(t) - p_j^i(t)\} \\ &\geq \max_j \{a_{ij}(t_\infty) - B(W + (V+1)(t - t_\infty)) - p_j(t_\infty)\} \\ &= -B(V+1)(t - t_\infty) + \left[\max_j \{a_{ij}(t_\infty) - p_j(t_\infty)\} - BW\right] \end{aligned} \tag{20}$$

which is also a decreasing affine function of time after t_0. Since the slope of the line described in (19) is less than the slope of the one described in (20), 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 $\bigcup_{i \in I^\infty} A(i) = J^\infty$. In a forward auction, once tasks are assigned they remain assigned, so 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. Since the algorithm terminates, the final assignment has a collective benefit within $m\epsilon$ of the maximum possible (with respect to the configuration of the agents at the time of termination) given the results of Lemmas 1 and 2 and Theorem 1. □

Here we note a few important things concerning the previous theorem. First, it should be clear that in the more general formulation with values B_{ij} for each agent-task pair instead of a single common value B, the validity of the proof is maintained by simply replacing B with $\max_{i,j} B_{ij}$. Secondly, the assumption that no agent reaches a task it is tracking prior to the termination of the algorithm is virtually impossible to guarantee for cases in which the communication delay between agents is large in relation to their speed. Thirdly, the value required for the bidding increment ϵ tends to be extremely large, and in practice (see Section V) forces the auction to conclude quickly and with poor results (particularly for a large number of agents since the sub-optimality bound $m\epsilon$ increases quadratically with m). The reason ϵ must be so large is that Theorem 2 requires an analysis of the worst case scenario. In this scenario, the agents are all bidding (and moving towards) a group of $m-1$ tasks that lie directly in front of them while moving directly away from the remaining tasks. The benefits of the first group only decrease when their

prices rise, while the benefits of the latter continually decrease as the agents move away from them at the fastest possible rate.

For these reasons we would like to guarantee the termination of algorithm under less restrictive conditions. The inequalities of (14) and (15) were based on the assumption that an agent's trajectory could lead anywhere so long as it was feasible. Considering instead a case where the agent's movement is restricted to a subset of the environment that includes all the tasks. Then for most situations the length of the optimal trajectory from an agent to a task should satisfy

$$0 \leq |\sigma[s_i(t), d_j]| \leq X \qquad \forall\, t > 0,\, \forall\, i, j \tag{21}$$

for some positive scalar X. Thus the associated benefit of that pair satisfies

$$C_{ij} - BX - Bt \leq a_{ij}(t) \leq C_{ij} - Bt \qquad \forall\, t > 0,\, \forall\, i, j \tag{22}$$

Theorem 3: If the motion of the agents is restricted in such a way that the inequality in (22) holds, then the auction algorithm terminates for any $\epsilon > 0$.

Proof: Without loss of generality, assume that no tasks are completed before a full assignment is reached. If they are handled correctly (see the next section), then an early task completion simply changes the assignment problem to another with one less agent and task.

In Theorem 2 we have already established that the price of every task in a partial assignment decreases by at least ϵ every $2D(m-1)$ time units. Then for some agent i, the value it associates with its preferred task from the current partial assignment $S(t)$

$$v_i(t) = \max_{j \in S(t)} \{a_{ij}(t) - p_j(t)\}$$

is upper bounded by

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

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 $t \to \infty$ if a full assignment is not reached, it is clear that after some finite time period, $v_i(t)$ will fall below $\overline{v}_i(t)$ and some task will be added to the assignment. This process must repeat until the assignment is full, therefore causing the algorithm to terminate in finite time. □

Note that the condition stated in (21) can be satisfied by a number of simple motion control schemes, including those laid out in Section IV-B. In this case, however, we loose the optimality bounds guaranteed from maintaining an $\epsilon-$CS condition since Lemmas 1 and 2 do not hold when an agent can reach a task prior to termination. As we will see in Section V, the choice of the bidding increment ϵ will have a large effect on the practical performance of the algorithm as it will determine the trade off between the advantage of acting quickly (to prevent too much degradation in collective benefit) and the advantage of optimizing the assignment with respect to the benefits at the time of termination.

D. Optimality Bounds

In this section we derive worst case bounds for the collective benefit received from the assignment reached at time t_a. Since we know that the collective benefit of any assignment will degrade as a function of the time it takes to achieve a full assignment, we start by considering a worst case bound on t_a. For the algorithm outlined in Section IV-B, we can use the intersection of the lines described in (19) and (20) as a guide for determining this value. Consider the m^{th} task to enter the assignment and let t_a be the time that one of the agents bids for it (since all agents will be tracking their final assignment from that time forward). The value $\overline{v}(t)$ of the final task to any agent is lower bounded by

$$\overline{v}(t) \geq \min_{i,j} a_{ij}(0) - B(W + (V+1)t)$$

and the value any of the agents places on any object already in the assignment must be upper bounded by

$$v(t) \leq \max_{i,j} a_{ij}(0) - \delta B(V+1)t + \epsilon$$

with t_a given as the intersection of the two lines defined by the right hand sides of the previous two equations because after that point, some unassigned task must be the preferred task for all agents (one of which must enter a bid). For the algorithm with motion control, the termination time t_a therefore can be bounded from above as

$$t_a \leq 2D(m-1)\frac{\max a_{ij}(0) - \min a_{ij}(0) + BW + \epsilon}{\epsilon - 2DB(V+1)(m-1)} \tag{23}$$

For comparison, we state a similar bound for the termination of a distributed auction with fixed benefits (the benefits "frozen" at their initial values) adapted from the results of [14] and [15]. In this case the termination time is bounded from above by

$$t_a \leq 2D(m-1)\left(\frac{\max a_{ij}(0) - \min a_{ij}(0)}{\epsilon} + 1\right) \tag{24}$$

where it is apparent that the worst case termination time for an auction performed with static benefits is always better than that of one in which the current values of benefits are used (given the same ϵ) . That said, a worst case collective benefit based solely on termination time and the related possible change in benefit given by (14) will always favor a fixed-benefit distributed auction [9]. We can, however, take another approach to give us an interesting result for the algorithm with motion control.

Let j_i denote the final task assignment of agent i and let $i^\star$ denote the agent who made the final bid and $j^\star$ the task receiving that bid. We are interested in finding a worst case lower bound for the collective benefit of that assignment, i.e.,

$$\sum_{i=1}^{m} a_{ij_i}(t_a) = a_{i^\star j^\star}(t_a) + \sum_{i=1, i\neq i^\star}^{m} a_{ij_i}(t_a) \tag{25}$$

where we have separated the benefit of the last assignment for the purpose of analysis. The lower bound for the benefit of this pair is given by the inequalities in (15) and (14) as

$$a_{i^\star j^\star}(t_a) \geq a_{i^\star j^\star}(0) - BW - B(V+1)t_a \tag{26}$$

Before returning to the other half of (25) we note two facts concerning the task prices at the moment of termination. First, the price of $j^\star$ is still equal to zero (it has received a bid, but that does not become its real price until the auctioneer responsible for $j^\star$ makes the assignment). 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 tasks must satisfy

$$\sum_{i=1,i\neq i^\star}^{m} p_{j_i}(t_a) \geq \epsilon \left\lceil \frac{t_a}{2D} \right\rceil \geq \epsilon \frac{t_a}{2D} \tag{27}$$

where $\lceil \cdot \rceil$ is the ceiling operator. Now consider the benefit terms under the sum on the right hand side of (25). Since the algorithm maintains an ϵ-CS condition for all assignments, we know that for each $i \neq i^\star$ the benefit of an agent's assigned task and the price of that task satisfy the following inequality

$$\begin{aligned} a_{ij_i}(t_a) - p_{j_i}(t_a) &\geq \max_k \{a_{ik}(t_a) - p_k(t_a)\} - \epsilon \\ &\geq a_{ij^\star}(t_a) - \epsilon \\ &\geq a_{ij^\star}(0) - BW - B(V+1)t_a - \epsilon \end{aligned} \tag{28}$$

where we have used the fact that the $p_{j^\star}(t_a) = 0$ and the lower bound on the possible change in benefit. Returning to the sum on the right hand side of (25)

$$\begin{aligned} \sum_{i=1,i\neq i^\star}^{m} a_{ij_i}(t_a) &= \sum_{i=1,i\neq i^\star}^{m} \Big(a_{ij_i}(t_a) - p_{j_i}(t_a) + p_{j_i}(t_a) \Big) \\ &= \sum_{i=1,i\neq i^\star}^{m} \Big(a_{ij_i}(t_a) - p_{j_i}(t_a) \Big) + \sum_{i=1,i\neq i^\star}^{m} p_{j_i}(t_a) \\ &\geq \sum_{i=1,i\neq i^\star}^{m} \Big(a_{ij_i}(0) - [BW + B(V+1)t_a + \epsilon] \Big) \\ &\quad + \sum_{i=1,i\neq i^\star}^{m} p_{j_i}(t_a) \end{aligned} \tag{29}$$

which allows us to start to determine a bound on the collective benefit obtained through the algorithm

$$\begin{aligned} \sum_{i=1}^{m} a_{ij_i}(t_a) \geq \; & a_{i^\star j^\star}(0) - BW - B(V+1)t_a + \sum_{i=1,i\neq i^\star}^{m} p_{j_i}(t_a) \\ & + \sum_{i=1,i\neq i^\star}^{m} \Big(a_{ij_i}(0) - [BW + B(V+1)t_a + \epsilon] \Big) \end{aligned}$$

For convenience we let $\underline{a} = \min_{i,j} a_{ij}(0)$ and $\overline{a} = \max_{i,j} a_{ij}(0)$, and let A^* denote the optimal collective benefit at time $t = 0$. Taking the minimum value for the benefit terms, the previous equation becomes

$$\begin{aligned} \sum_{i=1}^{m} a_{ij_i}(t_a) \geq \; & m\underline{a} - (m-1)\epsilon - mBW - B(V+1)t_a \\ & - B(V+1)(m-1)t_a + \sum_{i=1,i\neq i^\star}^{m} p_{j_i}(t_a) \end{aligned}$$

Using the lower bound for the sum of prices in (27) and the fact that $A^* \leq m\overline{a}$

$$\begin{aligned}\sum_{i=1}^{m} a_{ij_i}(t_a) \geq \quad & A^* - m(\overline{a} - \underline{a}) - (m-1)\epsilon - mBW \\ & -B(V+1)t_a - B(V+1)(m-1)t_a + \epsilon\tfrac{t_a}{2D}\end{aligned}$$

and rewriting ϵ in the last term as the product $\delta 2DB(V+1)(m-1)$, where $\delta > 1$ if the termination criteria of Theorem 2 is met, and $0 < \delta \leq 1$ otherwise.

$$\begin{aligned}\sum_{i=1}^{m} a_{ij_i}(t_a) \quad &\geq \quad A^* - m(\overline{a} - \underline{a}) - (m-1)\epsilon - mBW - B(V+1)t_a \\ & \qquad -B(V+1)(m-1)t_a + \delta B(V+1)(m-1)t_a \\ &= \quad A^* - m(\overline{a} - \underline{a}) - (m-1)\epsilon - mBW \\ & \qquad -[1-(\delta-1)(m-1)]B(V+1)t_a\end{aligned} \tag{30}$$

The bound in (30) is linear in terms of the termination time t_a which allows us to easily find its minimum on the interval $[0, \max\{t_a\}]$ by substituting the value for $\max\{t_a\}$ from (23) if $(\delta-1)(m-1) < 1$ or simply taking the value of the first four terms if otherwise. Since the bound provided in the latter case is extremely poor, we note that since the final assignment arrived at by the algorithm is guaranteed to be within $m\epsilon$ of the optimal solution at t_a combined with the maximum possible degradation of the optimal solution given the bounds on the change in benefits, we have a second lower bound given by

$$\sum_{i=1}^{m} a_{ij_i}(t_a) \quad \geq \quad A^* - m(BW + B(V+1)t_a) - m\epsilon \tag{31}$$

then the worst case benefit in the case where $(\delta-1)(m-1) \geq 1$ is given by the value of (30) and (31) at their intersection. Note that this value can be optimized by varying ϵ but, as simulations will show, the practical performance of the algorithm in most cases is significantly better than its worst case performance, so it will make little sense to optimize in this fashion.

As an example, we compare the worst case performance of the algorithm and a fixed-benefit distributed auction for the Dubins car model. For a generic situation involving this model, the best worst case bound for the latter algorithm is achieved by having the agents circle (loiter) until the auction is complete and is given by

$$\sum_{i=1}^{m} a_{ij_i}(t_a) \quad \geq \quad A^* - mB\left(\frac{2\pi R_{min}}{v_c} + t_a\right) - m\epsilon \tag{32}$$

For a given set of problem parameters ($m = 10$, $\overline{a} - \underline{a} = 100$, $B = 1$, $R_{min}/v_c = 0.1$) and with δ set to $1 + \frac{1}{m-1}$ (see preceding discussion) we plot the worst case bound as a function of termination time for both the algorithm with motion control and the fixed-benefit distributed auction over the interval $t = 0$ to $t = 2D(m-1)\frac{\overline{a}-\underline{a}}{\epsilon}$ (the upper bound of the termination time of the fixed-benefit distributed auction). The left plot in Figure 1, for a maximum delay of $D = 1$, clearly shows that the worst case performance of the fixed-benefit distributed auction (given by the minimum of the dashed line) is better than that of the algorithm with motion control. The right plot shows the

same case for a maximum delay of $D = 10$; the worst case bound of both algorithms has decreased, but that of the fixed-benefit distributed auction has now fallen below that of the one with motion control.

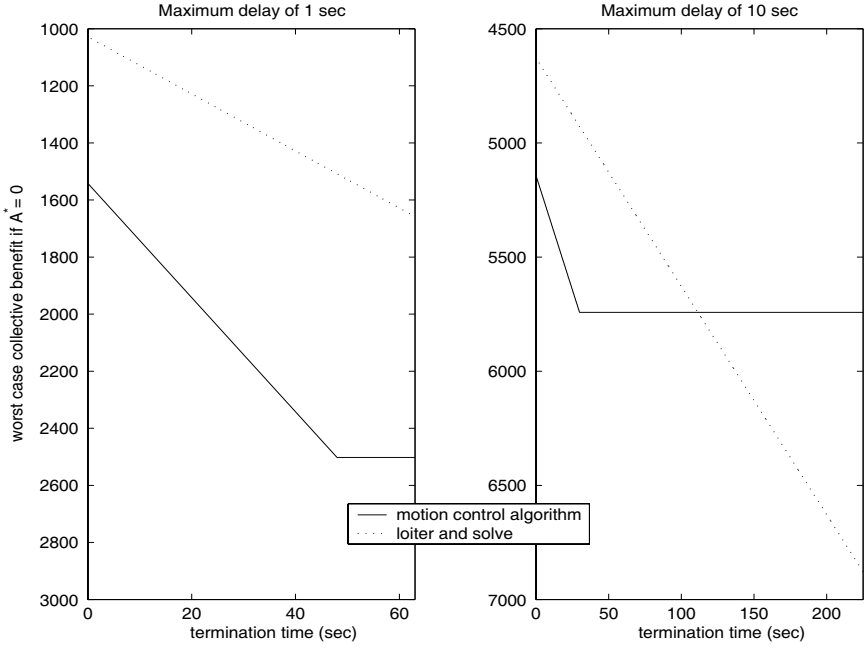


Fig. 1. Worst case comparison between algorithms.

E. Early Task Completion

The preceding analysis was all based on the assumption in Theorem 2 that no agent reaches a task prior to the termination of the algorithm. Since this is difficult (if not impossible) to guarantee in many situations, we are curious about how much the maximum possible benefit can degrade if we let agents complete a task prior to the termination of the algorithm. Intuition tells us that for cases where the benefits are highly dependent on the time of task completion (i.e., B large relative to $\max_{i,j} C_{ij} - \min_{i,j} C_{ij}$) there is more to gain from completing a task early than waiting for the algorithm to arrive at a full assignment. For a potentially large class of vehicle models, this turns out to be the case.

Theorem 4: Consider the configuration of agents and tasks at a specific instant in time. If

1) The vectors describing the position of the agents (s_i, $i \in \{1, \ldots, m\}$) and those describing the position of the tasks (d_j, $j = \{1, \ldots, n\}$) all belong to the same state space M,
2) The optimal trajectory distance function $|\sigma| : M \times M \to \mathbb{R}_{\geq 0}$ satisfies $|\sigma[s_i, d_j]| = |\sigma[s_{i'}, d_{j'}]|$ whenever $s_i = s_{i'}$ and $d_j = d_{j'}$, 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 $(i^\star, j^\star)$.

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 $A \leq A^\star$.

Assume $(i^\star, j^\star) \notin S$ (if $(i^\star, j^\star) \in S$, the hypothesis is trivial). Let i' be the agent assigned to $j^\star$ and j' the task assigned to $i^\star$ under S. For convenience, define the subset of S that contains these assignment pairs as $S' = \{(i^\star, j'), (i', j^\star)\}$. Now, using the optimal trajectory distance function we can state the value A as

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

where we use the assumption that $s_{i^\star} = d_{j^\star}$ and the fact that the optimal trajectory distance function must satisfy both the triangle inequality and $|\sigma[x, x]| = 0$. The final inequality follows from the fact that the prior sum is the total travel time of an assignment that includes the pair $(s_{i^\star}, d_{j^\star})$, which, by definition, is lower bounded by $A^\star$. Thus $A \leq A^\star \leq A \Rightarrow A^\star = A$, proving that the constrained assignment achieves the optimal value. □

The above theorem demonstrates that letting an agent complete a task early cannot degrade the maximum achievable benefit in problems where $C_{ij} = 0 \ \forall \ i, j$ (or when $C_{ij} = C_{kj} \ \forall \ i, j, k$ in symmetric assignment problems) provided that agents and tasks are defined as points in the same space and the optimal trajectory distance function has the stated properties. Note that if the speed and turning radius of each agent is identical, the Dubins Car model satisfies these criteria. The pivoting robot model, on the other hand, does not meet the first assumption of Theorem 4 except in the case where $\omega_{max} = \infty$.

Note that in the more general situation where the fixed portion of the benefit terms C_{ij} varies widely among agents, the early completion of a task $j^\star$ by agent $i^\star$ 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. Unless the agents have the ability to sense the status of a task before committing themselves, they must have a mechanism to avoid possible duplication. One manner of accomplishing this is to have auctioneer responsibility for a task fall on the last agent to which that task was assigned (with responsibility for an unassigned task remaining with the agent at which it started).

V. SIMULATIONS

In this section we present results from simulations using the modified auction algorithm from Section IV and agents with vehicle dynamics described by a Dubins car model. We do not attempt to characterize the algorithm's behavior in terms of all the salient parameters since there are a large number of them, many of which are interdependent (i.e., vehicle dynamics, communication methodology, number of agent/tasks, initial configurations, benefit range, weighting of benefit terms, etc.). Instead, we have selected a few scenarios that best illustrate the more important aspects of the problem. All simulations with the exception of the last one used 10 agents, since simulation time quickly became prohibitive with more, and 10 tasks since assignment problems are generally more difficult to solve when agents do not have the option of ignoring some tasks. In the cases where we compare the algorithm of Section IV to the fixed-benefit distributed auction, we sought to use an implementation of the latter that was proper for the scenario involved.

A. Effects of Delays

As previously discussed, the presence of communication delays will generally lengthen the time it takes the agents to reach an assignment regardless of the algorithm employed. The associated delay in getting the agents to the tasks will therefore have a detrimental impact on the collective benefit that is achieved. In order to assess the effects of communication delays, we simulated a situation in which the tasks are randomly distributed over one area and the agents over another area some distance away from the first (with agents initially headed towards the general area of the targets). See Figure 2 for an example of this configuration. For this scenario, the agents were given a speed of 100 m/s and a turning radius of 1000 m. The static benefit of the task ranged from 0 to 100 and the time dependent term entered at a rate of 1/sec which put the static and the initial time dependent terms of the benefit equations on roughly the same scale for this setup. A bidding increment of 25 was chosen as a value that ensured the algorithms terminated prior to an agent reaching a task most of the time. Early completion of tasks was allowed under the method of trading auctioneer duty discussed in Section IV-E. Random but bounded communication delays were simulated using a uniform distribution over a specified range. The maximum communication delay was varied from 0.5 sec to 10 sec for one case in which the minimum delay was zero and another in which the minimum delay was 80% of the maximum.

Figures 3 and 4 clearly show how the average benefit decreases with increased delay for both the algorithm with motion control and a fixed-benefit distributed auction in which the agents move along their initial heading towards the tasks until a full assignment is reached (labeled "solved while cruising" in the figures). They also demonstrate that the motion control algorithm tends to achieve a better collective benefit in comparison to the fixed-benefit distributed auction. This is particularly the case when the average delay is high, as shown in Figure 4. With large delay, the average benefit achieved with the fixed-benefit auction falls below even that of the average random assignment. We note here that the error bars on all the figures in this section are based on the standard deviation of the estimate of the mean (i.e., $\sigma = S/\sqrt{k}$ where S^2 is the sample variance and k the number of simulation runs).

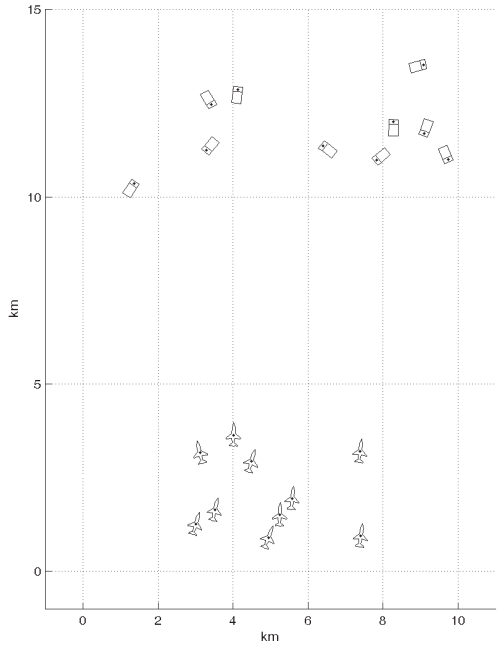


Fig. 2. Example of scenario for investigating effects of delays. Rectangles and planes denote tasks and agents respectively.

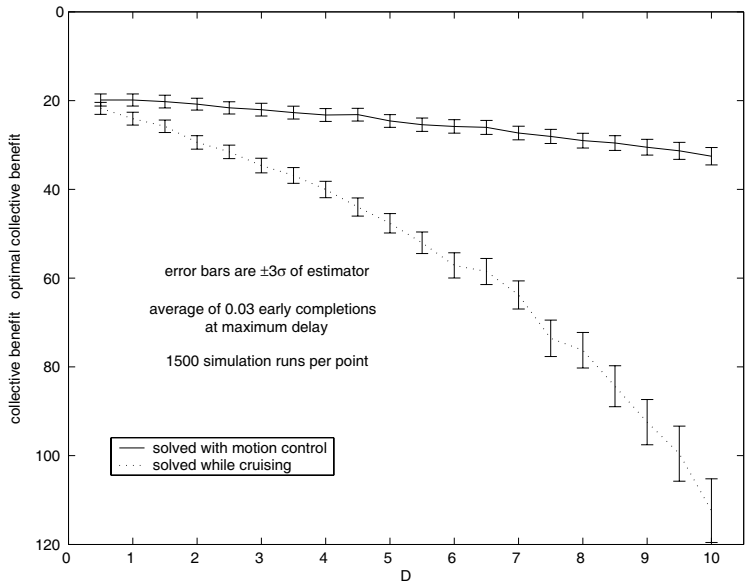


Fig. 3. Comparison of algorithms when delay is uniform on $[0, D]$.

B. Effects of Bidding Increment

As discussed in Section IV-C, in most situations it is virtually impossible to guarantee the assumption of Theorem 2 that no agent will reach a task before termination of the algorithm. On the other hand, the bidding increment specified in that theorem is often so large as to effectively eliminate the optimization process of the auction (i.e.,

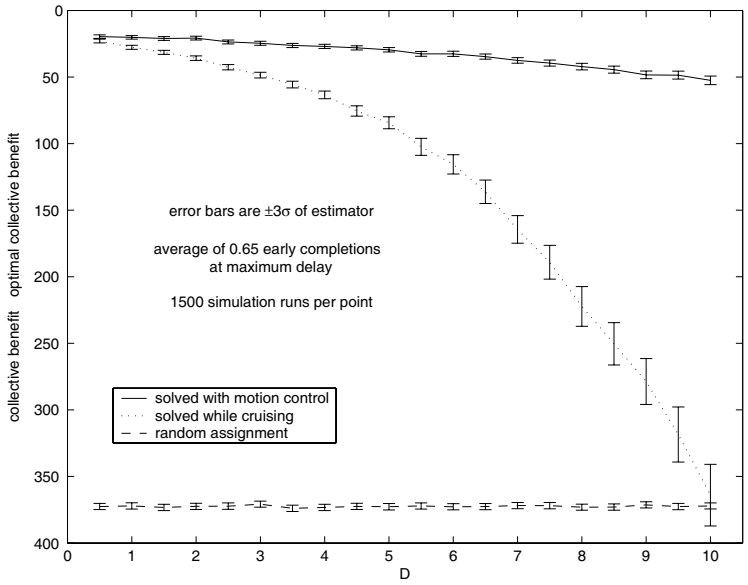


Fig. 4. Comparison of algorithms when delay is uniform on $[0.8D, D]$.

the first bid a task receives will raise its price so high that it is unlikely to receive another bid). In order to study the effects of varying the bid increment, a scenario similar to that in Figure 2 was used. The area covered was approximately twice as large and the agents' speed was increased to 300 m/s while its turning radius decreased to 500 m. These changes reduced the variance of the simulations, allowing us to highlight the effect of small changes in the bidding increment without having to run an excessive number of trials.

As Figure 5 shows, the best average performance of the algorithm for a given delay lies at a certain value of the bidding increment. As the bidding increment decreases from that value, the average benefit falls off steeply since it takes more time to reach an assignment. As the bidding increment increases from that value, the average benefit decreases as the algorithm is not attempting to achieve as much optimization. As is apparent from Figure 5, the optimal bidding increment also tends to decrease as the communication delay decreases. The balance between the competing goals of acting quickly and optimizing the assignment according to the current benefits is determined by the size of the bidding increment; a large value emphasizes the former and a small value the latter. When the communication delay decreases, the algorithm runs faster with respect to the changes in benefits, thereby tilting that balance in favor of performing a better optimization.

C. Early Task Completions

In Theorem 4 we saw that letting an agent complete a task early cannot decrease the achievable collective benefit in a total path minimization problem when a few simple properties hold for the length of the optimal trajectory between two points. To illustrate this we simulated a total path minimization scenario for a configuration of agents and tasks in which both are randomly distributed on a disc with radius twice the minimum turn radius of the

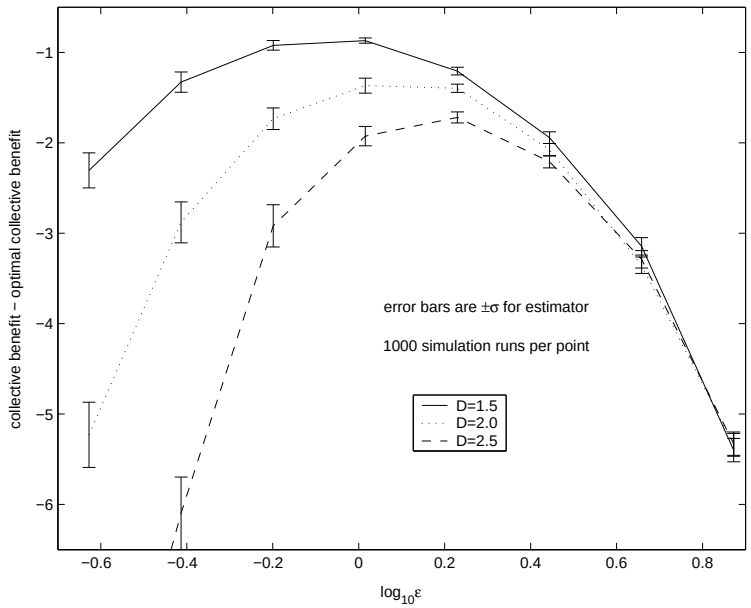


Fig. 5. Average performance as a function of the bidding increment (delays uniform on $[0.8D, D]$).

vehicles (see Figure 6). For this simulation the speed of the agents was 100 m/s and their turning radius 1000 m. The bidding increment was kept constant at 25 and the maximum communication delay again varied from 0.5 sec to 10 sec. One simulation was run in which agents where allowed to complete tasks early, and another in which they were forced to circle back to the task until the algorithm terminated (or their preferred task changed). Both cases were compared to a fixed-benefit distributed auction based on initial benefits in which the agents moved on a circle passing through their initial position (they "loiter") until the auction was complete.

The results appear in Figure 7. As expected, the motion control algorithm in which agents were allowed to complete tasks early performed much better than both of the other algorithms and is the only algorithm whose performance stays above that of a random assignment for the given values of D. The performance of the two motion control algorithms starts close together, but quickly diverges as communication delays increase. This is expected since the close proximity of agents and tasks will result in more and more agents having to pass tasks as the time it takes to reach an assignment increases. In fact, as the delays increase, the performance of the motion control algorithm without early task completions starts to resemble that of the fixed-benefit distributed auction.

D. Distributed vs. Centralized Computation

Given that the number of agents and tasks involved in many of these problems is relatively small, solving the associated assignment problem on a single processor can take considerably less time than implementing a distributed auction that is prone to protracted "price wars" (i.e., agents making small incremental bids for a few tasks and incurring a communication delay with each bid and assignment). In order for the motion control algorithm to be practical, it would have to be the case that the benefits of its parallel computation exceeded the communication

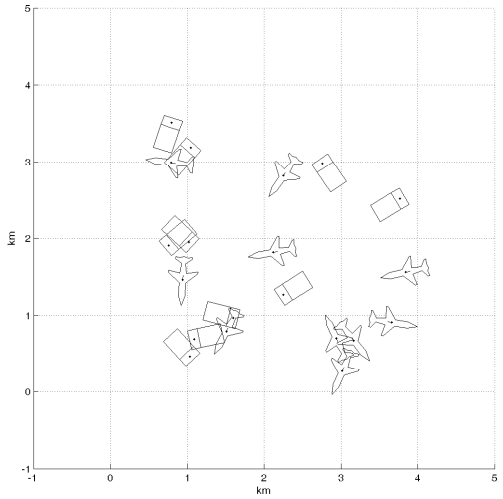


Fig. 6. Example of total path minimization problem.

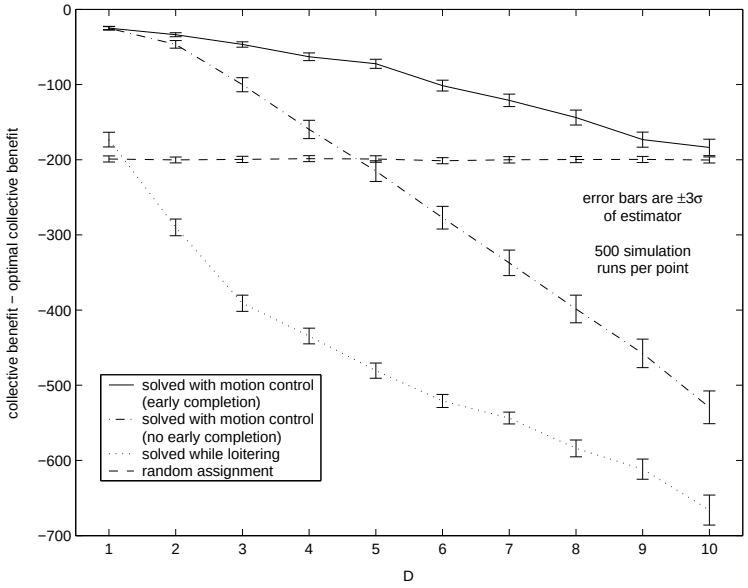


Fig. 7. Comparison of algorithms in total path minimization problem (delays uniform on $[0.8D, D]$).

penalty associated with the bidding process. If we are given a communication network over which all the problem data could be efficiently sent to every agent and the assignment problem solved redundantly, it would be hard to justify using a distributed methodology. However, as an example of when the algorithm of Section IV-B may prove more effective, consider the case where communications are limited by bandwidth. In a scenario where the structure of the benefits tends to result in the modified auction taking only a few iterations, then that algorithm might tend to terminate faster (and produce a higher collective benefit) than it takes for the agents to transmit all the problem data to each other.

The total path minimization problem is just such a scenario; if the agents and tasks are distributed in the same manner across a wide area, most agents will wind up assigned to the task to which they are closest initially. For this simulation, we used a time division multiplex (TDM) scheme for the communication network, with the time slot allocated to each agent the size of a distributed auction message, denoted as T. To approximate the time it would take to transmit all the problem data to the agents, we roughly estimate the time required for a single field to be $\frac{1}{4}T$ since an auction message has four fields (i.e., task, agent, bid/price, and a time stamp). Then in order to synchronize their data each agent would have to send a message of $2+n$ fields (time stamp, agent, and its benefit for each of n tasks) if the tasks can be placed in a known order or $2+2n$ fields otherwise. Then transmitting the entire problem data for every agent would require $\frac{1}{4}Tm(2+n)$ in the first case (scheme 1) and $\frac{1}{4}Tm(2+2n)$ in the second (scheme 2).

For this simulation we distributed the agents and tasks randomly over a 20 km square, with the speed and turning radius of the agents again at 100 m/s and 1000 m respectively. In both "synchronized schemes," the agents move towards their closest initial task while they wait to collect all the benefit data. Once that is complete, they proceeded towards their assignment from the optimal solution. The results appear in Figure 8 and clearly show that there are crossover points at which the average performance of the distributed auction with motion control becomes greater than that of the two synchronized schemes as the bandwidth of the communication network decreases. Hence, we can conclude that with a poor quality communication network the distributed auction with motion control can outperform a centralized algorithm. We note that the performance of the synchronized schemes appears to degrade linearly with decreasing bandwidth (with slopes roughly proportional to their synchronization time), while the performance of the motion control algorithm becomes decidedly nonlinear for low bandwidth values in such a way that cannot be attributed to the variance of the simulation data. While we do not have a proper explanation for this phenomenon, we assume it is the effect of some interaction between the switching behavior of the motion control algorithm, the synchronicity imposed by TDM communication, and the geometry of the task scenario. The practical advantages of the motion control algorithm in this scenario are even more evident (and without when the number of tasks is large relative to the number of agents (see Figure 9). The additional tasks do not generally make these small assignment problems more difficult to solve (since we can still assume calculation times are negligible), but the communication requirements of the synchronized schemes quickly cause their performance to degrade as the available bandwidth shrinks.

VI. Conclusions

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 [9] 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.

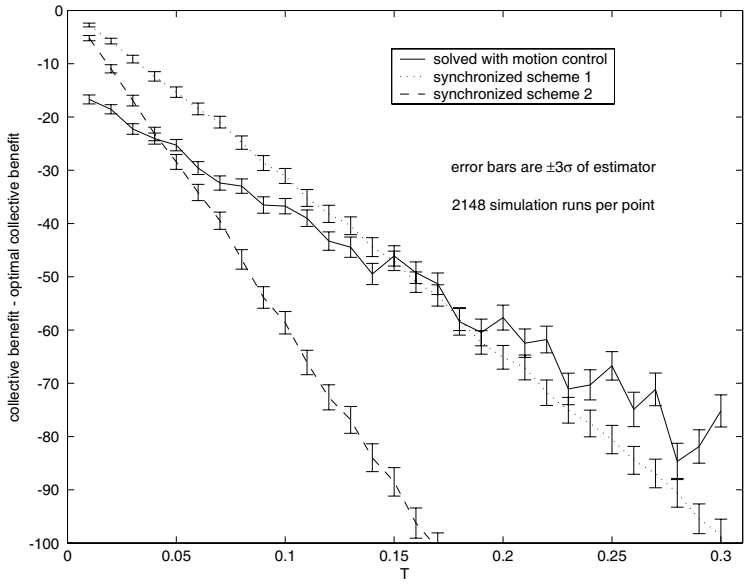


Fig. 8. Comparison of algorithms in bandwidth constrained scenario (10 agents/10 tasks).

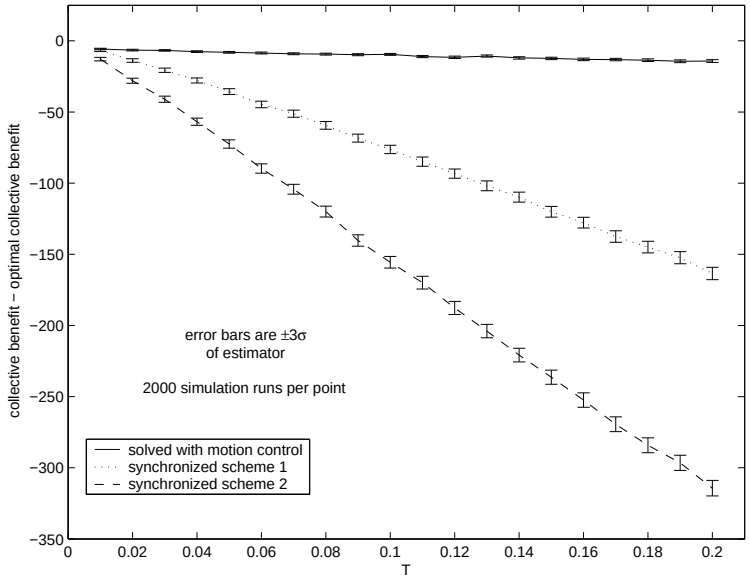


Fig. 9. Comparison of algorithms in bandwidth constrained scenario (10 agents/20 tasks).

Simulations demonstrated that the motion controlled algorithm has an average performance superior to that of a fixed-benefit distributed auction in many cases, and may even outperform a centralized approach in certain situations.

There are two key extensions of the studied problem that deserve further research. One of these is a multi-assignment version in which agents are capable of completing more than one task. In this case, an agent's trajectory to a task (and the benefit it will receive from it) is directly related to the other tasks that agent must visit. Another

case that presents similar problems is a version in which tasks are coupled to each other (i.e., certain tasks cannot be completed before others). Here, the benefit an agent will receive from a task may depend on what other agents are doing and when. In both these cases the non-linear coupling between tasks and benefits rules out algorithms, such as the auction, that are designed for linear network flow models. Given that these problems often take a longer time to solve, the concept of directing agents' motion during that process would seem to be of even more importance.

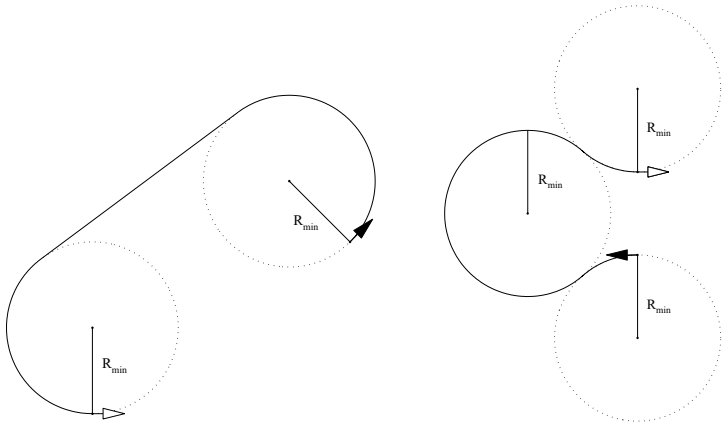
APPENDIX

Many vehicles are forced to travel at (or within a narrow range of) a constant velocity while possessing only limited steering ability for one reason or another. Cruising aircraft, for example, commonly achieve the greatest fuel efficiency at a specific airspeed and can only turn so fast before they start losing altitude. The speed of an automobile on a busy highway, on the other hand, is largely dictated by the flow of traffic with the driver unable to execute too sharp a turn without losing control or rolling over. When motion is confined to a plane as it is in these two cases, we can model the vehicle kinematically as what is known as the Dubins car [4] model consisting of the system of non-linear differential equations and control input constraints given in (12)

Let us define the state vector of this system as $s = [x, y, \theta]^\top$. We then denote a trajectory in this state space as $s(t), t \geq 0$ and a particular point on such a trajectory as $s(t)$. We say that a particular trajectory $s(t), t \geq 0$ is feasible if there exists a control input $u(t), t \geq 0$ satisfying the constraint $|u(t)| \leq 1$, such that $s(t), t \geq 0$ is a solution to the system in (12). Let us denote the projection of $s(t), t \geq 0$ into the xy-plane as the parameterized curve $s_{xy}(t)$. For all feasible trajectories, the curve $s_{xy}(t)$ possesses the following properties: 1) the vehicle is always oriented in the same direction as the tangent to $s_{xy}(t)$, 2) the length of any segment of $s_{xy}(t)$ is proportional to the time taken to traverse it, so distance optimal paths are also time optimal, and 3) the magnitude of the trajectory's radius of curvature is bounded from below by the value $R_{min} = v_c/\omega_{max}$.

In [4], Dubins showed that for this model the shortest feasible trajectory between two points (with initial and final headings specified) must be constructed solely from straight line segments and arcs with radius of curvature $\pm R_{min}$.Moreover, optimal trajectories must belong to one of two classes of curves. The first class, denoted *CSC*, consists of curves beginning with an arc of radius R_{min}, followed by a straight line segment, followed by another arc of radius R_{min} (where the turn direction of each arc is not restricted). The second class, denoted *CCC*, consists of curves comprised of three arcs of radius R_{min} (and alternating turn direction) in succession. Figure 10 provides an example trajectory of each type, illustrating their relationship to the the vehicle's turn circles (the two circles of radius R_{min} tangent to the vehicle's path) at the initial and final positions. Obviously, trajectories of the types *CC*, *CS*, *SC*, *C*, and *S* are simply subpaths of trajectories from the two main classes.

From this point on, we will assume that $s(t), t \geq 0$ is a feasible but otherwise arbitrary trajectory. At different points $s(t)$, the optimal trajectory to a fixed final destination and heading $d = [x_d, y_d, \theta_d]$ will vary. As before, we denote the optimal trajectory between these two points as $\sigma[s(t), d]$, and define the distance function $|\sigma[s(t), d]|$ as the travel time of $\sigma[s(t), d]$ (since velocity is constant, $|\sigma[s(t), d]|$ is just the length of $\sigma[s(t), d]$ divided by v_c). For the purposes of this work, we wish to find bounds on $|\sigma[s(t), d]|$ in the form of the following inequality,



(a) *CSC* curve (b) *CCC* curve

Fig. 10. Examples of the two classes of optimal trajectories (black and white arrows denote initial and final positions respectively).

$$|\sigma[s(t+\Delta t),d]| - |\sigma[s(t),d]| \leq V\Delta t + W \quad \forall\ t, \forall \Delta t > 0 \tag{A.1}$$

There are multiple methods that could be used to find a bound of this form. One approach might be to find an upper bound on $\frac{d}{dt}\,|\sigma[s(t),d]|$ everywhere $|\sigma[s(t),d]|$ is continuous. We would then have a suitable value for V provided we can find a value for W large enough to account for the maximum cumulative effect of possible discontinuities. Here, we instead take a geometric approach that establishes the maximum range for $|\sigma[s(t),d]|$ at an arbitrary point on $s_{xy}(t)$ and then uses this information to deduce V and W.

Theorem A.1: For an arbitrary trajectory $s(t), t \geq 0$ satisfying the feasibility conditions of (12), $|\sigma[s(t),d]|$ satisfies the inequality of (A.1) with $V = 1$ and $W = \frac{(2+3\pi)R_{min}}{v_c}$.

Proof: Since the basic spatial relationship between a vehicle and a target destination is preserved by the operations of translation and rotation, we can assume that the target destination is located at the origin and has heading pointing along the positive x-axis ($d = [0,0,0]^\top$) without any loss of generality. We first note that of the six trajectory types, a *LRL* and/or *RLR* trajectory may not exist when the vehicle is far away from the target position. Similarly, *LSR* and/or *RSL* trajectories may not exist when the vehicle and target position are close. Trajectories of type *LSL* and *RSR*, however, always exist for any $s_{xy}(t)$ because of the geometry of their construction. Hence, we will focus on finding an upper bound for the length of the *LSL* and *RSR* trajectories in terms of the point $s_{xy}(t)$ (i.e., letting $\theta(t)$ vary), which we can then use as a conservative bound on $|\sigma[s(t),d]|$. The diagram in Figure 11 illustrates the construction of these trajectories and the information we need to find this bound.

Consider the line segments D_L, Q_L, and R_{min}, where D_L is the same length as the straight segment in the *LSL* trajectory. Because they form a triangle, we have the inequality $D_L \leq Q_L + R_{min}$. But Q_L, $||s_{xy}(t) - (0,0)||_2$, and R_{min} also form a triangle, so $Q_L \leq ||s_{xy}(t) - (0,0)||_2 + R_{min}$. Therefore $D_L \leq ||s_{xy}(t) - (0,0)||_2 + 2R_{min}$, giving us an upper bound on the length of the *LSL* trajectory's middle segment (where D_L reaches this bound only

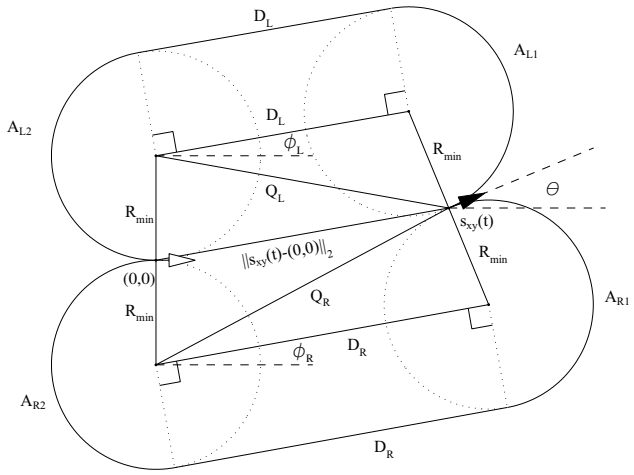


Fig. 11. Construction of *LSL* and *RSR* trajectories.

when the initial point lies on the negative y-axis with initial heading equal to π radians). By a similar argument, the same bound exists for the middle segment of the *RSR* trajectory.

Now we must account for the possible contribution of the arc segments to each trajectory length. This can easily be bounded by $4\pi R_{min}$ (two arcs of maximum length $2\pi R_{min}$ each), but we are motivated to find a tighter bound in order to improve the final result of this proof. Consider the sum of the length in radians of all the arcs from both the *LSL* and *RSR* trajectories, which we denote C_T. Using some elementary geometry we have

$$\begin{aligned} C_T &= \underbrace{(\pi + \phi_L - \theta) \bmod 2\pi}_{A_{L1}} + \underbrace{(\pi - \phi_L) \bmod 2\pi}_{A_{L2}} \\ &\quad + \underbrace{(\pi - \phi_R + \theta) \bmod 2\pi}_{A_{R1}} + \underbrace{(\pi + \phi_R) \bmod 2\pi}_{A_{R2}} \end{aligned}$$

which can be rewritten using the definition of the modulus operator ($a \bmod b = a - b\lfloor \frac{a}{b} \rfloor$, where $\lfloor \cdot \rfloor$ is the floor operator) to demonstrate that $C_T \in \{0, 2\pi, 4\pi, 6\pi\}$ for all $\phi_L, \phi_R, \theta \in [0, 2\pi)$, so that the total length of the arc segments in both trajectories is upper bounded by $6\pi R_{min}$.

To formulate an upper bound for $|\sigma[s(t), d]|$, we can use the length of either the *LSL* or *RSR* trajectory. Since they both always exist, we will always choose the one with the smaller total arc length. The length of the chosen trajectory can be no more than the maximum length of the middle segment, $||s_{xy}(t) - (0,0)||_2 + 2R_{min}$, plus half the total combined arc length of the two trajectories, $3\pi R_{min}$ (if the arc length of the chosen trajectory exceeds this amount, it would not be the minimum as we have established that the total arc length does not exceed $6\pi R_{min}$). Thus we have

$$|\sigma[s(t), d]| \leq \frac{||s_{xy}(t) - (0,0)||_2 + (2 + 3\pi)R_{min}}{v_c} \tag{A.2}$$

To establish a conservative range for $|\sigma[s(t), d]|$ at the point $s_{xy}(t)$, we will also need a lower bound. The length of $\sigma[s(t), d]$ must obviously be at least equal to the straight line distance between the point $s_{xy}(t)$ and the origin,

so we have

$$|\sigma[s(t),d]| \geq \frac{||s_{xy}(t)-(0,0)||_2}{v_c} \tag{A.3}$$

We will need one more inequality before we are ready to prove the final result. We can see that by the triangle inequality,

$$||s_{xy}(t+\Delta t)-(0,0)||_2 \leq ||s_{xy}(t)-(0,0)||_2 + ||s_{xy}(t+\Delta t)-s_{xy}(t)||_2 \tag{A.4}$$

but since the trajectory must be feasible, the path length between $s_{xy}(t)$ and $s_{xy}(t+\Delta t)$ is fixed by the time interval Δt. Accordingly, the straight line distance between the two points can be no more than $v_c\Delta t$ and (A.4) becomes

$$||s_{xy}(t+\Delta t)-(0,0)||_2 \leq ||s_{xy}(t)-(0,0)||_2 + v_c\Delta \tag{A.5}$$

We can now derive an inequality of the desired form. The increase between $|\sigma[s(t),d]|$ and $|\sigma[s(t+\Delta t),d]|$ can be upper bounded by the difference between the largest possible value of $|\sigma[s(t+\Delta t),d]|$ and the smallest possible value of $|\sigma[s(t),d]|$. By first applying the bounds in (A.2) and (A.3), then using the triangle inequality of (A.5), we can finally show that

$$\begin{aligned}
|\sigma[s(t+\Delta t),d]| - |\sigma[s(t),d]| &\leq \max_{s(t+\Delta t)} |\sigma[s(t+\Delta t),d]| - \min_{s(t)} |\sigma[s(t),d]| \\
&\leq \frac{||s_{xy}(t+\Delta t)-(0,0)||_2 + (2+3\pi)R_{min}}{v_c} \\
&\quad - \frac{||s_{xy}(t)-(0,0)||_2}{v_c} \\
&\leq \frac{||s_{xy}(t)-(0,0)||_2 + v_c\Delta t + (2+3\pi)R_{min}}{v_c} \\
&\quad - \frac{||s_{xy}(t)-(0,0)||_2}{v_c} \\
&= \Delta t + \frac{(2+3\pi)R_{min}}{v_c}
\end{aligned}$$

$$\Rightarrow \quad |\sigma[s(t+\Delta t),d]| - |\sigma[s(t),d]| \leq \Delta t + \tfrac{(2+3\pi)R_{min}}{v_c} \quad \forall t, \forall \Delta t > 0 \tag{A.6}$$

which fits the desired form of (A.1) with $V=1$ and $W=\frac{(2+3\pi)R_{min}}{v_c}$. □

The values for V and W arrived at in above are not unique in the sense that we might be able to derive other, equally valid pairs (V,W) for the same vehicle model that are not trivially different. Although not proven, simulation leads us to believe that we should be able to reduce W to possibly as little as $\frac{2\pi R_{min}}{v_c}$ if we are willing to accept a three-fold increase in V.

REFERENCES

[1] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conference*, (Anchorage, Alaska), pp. 1936–1941, May 2002.

[2] J. K. Howlett, T. W. McClain, and M. A. Goodrich, "Learning Real-Time A* path planner for sensing closely-spaced targets from an aircraft," in *AIAA Guidance, Navigation, and Control Conference*, no. 2003–5338, (Austin, Texas), August 2003.

[3] C. Schumacher, P. R. Chandler, M. Pachter, and L. Pachter, "Uav task assignment with timing constraints," in *AIAA Guidance, Navigation, and Control Conference*, no. 2003–5664, (Austin, Texas), 2003.

[4] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal position," *American Journal of Math*, vol. 79, pp. 497–516, 1957.

[5] D. H. Walker, T. W. McClain, and J. K. Howlett, "Coordinated uav target assignment using distributed tour calculations," in *4th Annual Conference on Cooperative Control and Optimization*, (Gainesville, Florida), December 2003.

[6] C. Schumacher, P. R. Chandler, and S. J. Rasmussen, "Task allocation for wide area search munitions via iterative network flow," in *AIAA Guidance, Navigation, and Control Conference*, no. 2002–4586, (Monterey, CA), 2002.

[7] A. E. Gil, K. M. Passino, S. Ganapathy, and A. Sparks, "Cooperative scheduling of tasks for networked uninhabited autonomous vehicles," in *42nd IEEE Conference on Decision and Control*, (Maui, Hawaii), pp. 522–527, December 2003.

[8] J. Finke, K. M. Passino, and A. Sparks, "Cooperative control via task load balancing for networked uninhabited autonomous vehicles," in *42nd IEEE Conference on Decision and Control*, (Maui, Hawaii), pp. 31–36, December 2003.

[9] D. P. Bertsekas and D. A. Castañon, "Parallel synchronous and asynchronous implementations of the auction algorithm," *Parallel Computing*, vol. 17, pp. 707–732, 1991.

[10] D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Bellmont, MA: Athena Scientific, 1997.

[11] D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Belmont, MA: Athena Scientific, 1998.

[12] D. A. Castañon and C. Wu, "Distributed algorithms for dynamic reassigment," in *42nd IEEE Conference on Decision and Control*, (Maui, Hawaii), pp. 13–18, December 2003.

[13] H. J. Sussman and G. Tang, "Shortest paths for the Reeds-Shepp car: a worked out example of the use of geometric techniques in nonlinear optimal control," Tech. Rep. SYCON-91-10, Rutgers University, New Brunswick, NJ, 1991.

[14] D. P. Bertsekas, "Auction algorithms for network flow problems: A tutorial introduction," Tech. Rep. LIDS-P-2108, Laboratory for Information and Decision Systems Report, M.I.T., Cambridge, MA.

[15] J. N. Tsitsiklis and D. P. Bertsekas, *Parallel and Distributed Computation: Numerical Methods*. Bellmont, MA: Athena Scientific, 1997.