REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**			
1. REPORT DATE *(DD-MM-YYYY)* 14-09-2005	**2. REPORT TYPE** Conference Paper Preprint		**3. DATES COVERED** *(From - To)* 2005
4. TITLE AND SUBTITLE Optical Dynamic Assignment for Low Earth Orbit Satellite Constellations			**5a. CONTRACT NUMBER**
			5b. GRANT NUMBER
			5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Alexander Melin, R. Scott Erwin*, VijaySekhar Chellaboina			**5d. PROJECT NUMBER**
			5e. TASK NUMBER
			5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES) Mechanical, Aerospace & Biomedical Engineering Dept University of Tennessee Knoxville, TN 37900	*Air Force Research Laboratory Space Vehicles 3550 Aberdeen Ave SE Kirtland AFB, NM 87117-5776		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory Space Vehicles 3550 Aberdeen Ave SE Kirtland AFB, NM 87117-5776			**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/VSSV
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			
13. SUPPLEMENTARY NOTES Submitted to 2006 American Control Conference, Minneapolis, MN. Government Purpose Rights			
14. ABSTRACT In this paper we investigate autonomous task assignment for a group of low-earth orbit satellites that don't necessarily have prior knowledge of the targets of interest. Specifically, we consider the optimal assignment problem for dynamic weighted bipartite graphs. First, we present necessary and sufficient conditions for the existence of a perfect matching in a given bipartite graph. Next, we present an algorithm to construct a *virtual* graph based on the original graph that guarantees the existence of a perfect matching. These results are then used to solve the optimal assignment problem for dynamic weighted bipartite graphs. Finally, we apply this algorithm to a constellation of low-earth orbit satellites.			
15. SUBJECT TERMS Linear Programming, Dynamic Assignment Problem, Dynamic Graph Topology, Dynamic Task Allocation, Graph Theory, Satellite Constellations			

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** R. Scott Erwin
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	Unlimited	8	**19b. TELEPHONE NUMBER** *(include area code)* 505-846-9816

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

Optimal Dynamic Assignment for Low Earth Orbit Satellite Constellations

Alexander Melin, R. Scott Erwin, and VijaySekhar Chellaboina

***Abstract*— In this paper we investigate autonomous task assignment for a group of low-earth orbit satellites, that don't necessarily have prior knowledge of the targets of interest. Specifically, we consider the optimal assignment problem for dynamic weighted bipartite graphs. First, we present necessary and sufficient conditions for the existence of a perfect matching in a given bipartite graph. Next, we present an algorithm to construct a *virtual* graph based on the original graph that guarantees the existence of a perfect matching. These results are then used to solve the optimal assignment problem for dynamic weighted bipartite graphs. Finally we apply this algorithm to a constellation of low-earth orbit satellites.**

***Index Terms*— Linear programming, dynamic assignment problem, dynamic graph topology, dynamic task allocation, graph theory, satellite constellations**

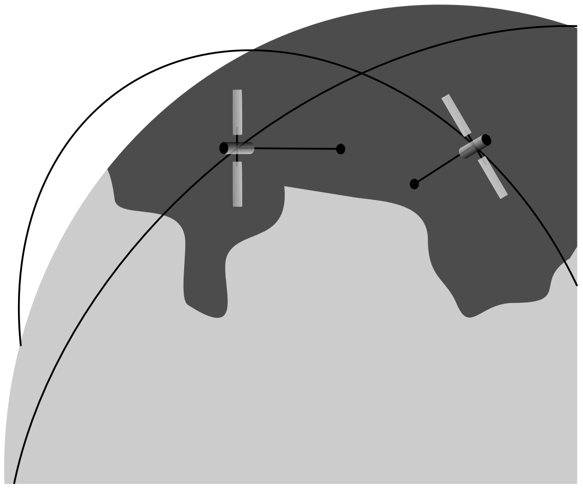

Fig. 1. Depiction of satellites tracking ground based targets

I. Introduction

In recent years there has been increased interest in the use of constellations of satellites and satellite formations. Constellations of satellites are used when global or nearly-global coverage are needed. Some applications include voice communication, satellite radio, broadband networking, remote sensing, and laser communications. Satellite formations, on the other hand, employ close range formations and the functionality of the satellites depends on the configuration of the formation.

Alexander Melin and VijaySekhar Chellaboina are with the Mechanical, Aerospace and Biomedical Engineering Department at the University of Tennessee, Knoxville, TN.

Scott Erwin is with the Air Force Research Lab, Albuquerque, NM.

A lot of research has been focused on the control of satellite formations. This is, in part, due to the novelty of the applications, such as the Terrestrial Planet Finder. For applications such as radar and imaging, the effective aperture of the satellite can be increased without a corresponding increase in the weight and size of the satellite due to the replacement of the majority of the structural elements with virtual structures. Much of the research has concentrated on controlling the geometry of the constellation and more recently on dynamic reconfiguration [1].

Controlling formation geometry lends itself to a simple graph theoretic interpretation. In general, a satellite constellation or UAV formation is considered as a directed graph where the nodes of the graph are the satellites or UAV's, the arcs of the graph are the communications channels and the physical position of the nodes corresponds to the physical position of the satellites or UAV's [2], [3].

On the other hand, satellite constellations such as GlobalStar do not rely on the relative formation to provide their functionality, but the specific orbits but the orbits the satellites are placed in. These constellations are generally used for global radio-frequency communication. RF communication can achieve global or near-global coverage with a relatively small number of satellites due to the wide coverage area of an individual satellite and overlapping footprints.

Laser communication and most remote sensing satellites have a narrow beamwidth so simultaneous global coverage would require a prohibitively large number of satellites. To achieve global coverage with narrow beamwidth satellites, one solution is to place the satellites into a global coverage constellation with overlapping footprints and assign the satellites to look at targets in their footprint in some an manner.

Currently, for many remote sensing systems, the assignment of targets to a satellite assumes that the target and satellite positions are known ahead of time. This allows a schedule to be calculated ahead of time and passed to the relevant satellites. In the case of laser communications this means that a pre-determined time window is scheduled for a connection. A more effective laser communications system would autonomously reconfigure itself to allow the user bandwidth-on-demand.

Performing this scheduling on the ground means that the system is not quickly adaptable to changes, in addition, delays are introduced by the need to recalculate and retransmit the assignments.

A. Problem Assumptions

This paper proposes a method based on the optimal assignment problem in linear programming that allows a group of satellites to autonomously change their task assignments in a distributed process. In order to make the problem more concrete, we will look at a general group of satellites and make some assumptions on their properties and abilities.

Specifically, we will look at groups of satellites whose tasks are to acquire and track specific targets on the surface of the earth for purposes of communication, imaging, etc. The satellites must have the ability to communicate with each other. Since we are concerned with high level task assignment, the satellites are assumed to have a stable attitude controller, and be able to track a point on the earth given the target's position.

We will assume that each satellite has the capability to perform any task assigned to it (i.e. track any target that is in range). Because of the overlapping nature of the satellites' abilities to perform the tasks, the objective of the dynamic scheduling problem is to keep satellites from trying to perform the same task simultaneously, while choosing the tasks that maximize the overall efficiency of the group.

A purely decentralized control architecture for this problem would require no communication between satellites, and each satellite would require knowledge of only its own internal states. We will assume an architecture which is predominantly decentralized but allows for minimal communication between the satellites, as follows: the satellite must be able to calculate three pieces of information and must transmit and receive one piece of this information.

- *i)* Each satellite must know all the potential targets and their locations.
- *ii)* Each satellite must be able to determine which targets are in range and which are out of range.
- *iii)* Each satellite must be able to calculate a performance metric that describes the efficiency of the satellite tracking each target in its own range.
- *iv)* Each satellite must be able to transmit this vector of performance metrics to all other satellites.

The control objective is to maximize the time each satellite spends tracking targets over an interval of time, subject to the limitations of the satellites' abilities. This problem also has many other similar applications such as missile tracking and can significantly reduce operator workload, and is especially valuable in time-critical military applications [4]. Because of the assumptions on the homogeneity of the satellites' abilities and the tasks to be assigned, the problem can be described by a graph theoretic approach, or an equivalent problem in linear programming known as the assignment problem.

B. Dynamic Graphs and the Assignment Problem

The assignment problem in graph theory and linear programming has been well understood since 1955 when Kühn used a theorem by the Hungarian mathematician König to create an algorithm, known as the Hungarian method, that finds an optimal perfect assignment [5]. A more precise definition of an optimal perfect assignment will be given later. Since this algorithm yields the exact solution to this problem, later work in this field is concerned with improving the speed of the algorithm [6], [7]. In graph theoretic notation, this problem is a matching on an edge weighted bipartite graph where one set of nodes corresponds to the satellites, the other set of nodes corresponds to the targets, the edges of the graph connect the satellites and targets that are in range, and the weights on the edges correspond to a performance metric for a satellite to track that target.

The solution to the optimal assignment problem assumes that the graph is static, however this problem cannot be described by a static graph. The number of satellites and targets are constantly changing, and the edges of the graph also change as the satellites orbit the earth and targets move in and out of range of satellites. Finally the edge weights are changing as the satellite's attitude changes to track a specific target.

Unfortunately very little is known about dynamic graphs [8], however, some research is being done in the area of combinatorics and graph theory applied to spacecraft formation flying [1], [9].

The problem of controlling large groups of autonomous vehicles has been studied in many forms. This is an extremely complex problem. Even finding a universal model for these distributed systems has not been accomplished [10]. One related problem is the scheduling problem. The most common uses for this formulation are in flexible manufacturing systems, computer networks, and processor resource allocation. In general this problem is NP-complete [11]. Some simplified cases have been solved in polynomial time, but due to the complexity of the problem heuristic methods are the preferred solution to this problem. The scheduling problem consists of allocating tasks or jobs to resources so that an optimality condition is satisfied. The most common optimality condition is minimizing the total job time [11], [12].

Another approach to controlling groups of autonomous vehicles is cooperative control. This approach is concerned with controlling the interactions between the elements of the system so that a goal is accomplished. The tasks studied in coordinated control generally require very complex interactions, such as RoboFlag [10]. Generally a ”best” solution to these problems cannot be determined. Most of the literature on cooperative control deals with languages to program the interactions between the robots so that the interactions are stable and heuristic strategy algorithms can be implemented to achieve the desired behavior. Many applications of cooperative control also use decentralized control as part of their control paradigm.

The problem most similar to this problem is the design of network routing protocols. The main difference being, that there is a cost associated in reconfiguring a satellite constellation, while reconfiguring a network does not involve any loss of performance. Reconfiguration of network is done mainly to prevent channel overloads and reconfigure in the event of a channel failure.

One main similarity in these approaches is that all the goals of the problem require a high level of interaction between the separate elements of the system for the goal to be accomplished. In flexible manufacturing, some jobs must be completed before others. In many instances a single agent cannot complete the goal. In cooperative control the formation and interaction between the agents is also necessary for the completion of the task. In autonomous formation flying, for example, the function of the spacecraft depends on the formation itself.

The organization of the remainder of the paper is as follows. In Section II we will introduce some notation and cover the concepts of graph theory required for the remainder of the paper. We also introduce the optimal assignment problem and discuss the solutions to it that exist in the literature, and prove the existence of a solution. In Section III we introduce the concepts of the dynamic graph and discuss how applying optimal assignment problem to a dynamic graph affects the behavior of the algorithm.

II. Mathematical Preliminaries

In this section, we introduce notation and some key results necessary for developing the main results of this paper. Let $\mathbb{R}$ (resp., $\mathbb{Z}$) denote the set of real numbers (resp., positive integers), let $\mathbb{R}^{n\times m}$ (resp., $\mathbb{Z}^{n\times m}$ denote the set of real (resp., positive integer) matrices, and let A^T denote the transpose of A.

Let $G = (V, E)$ denote a *bipartite graph* where $V = X \times Y$ is the set of vertices and E is the set of edges. It is convenient to assume that the elements of X and Y are enumerated for $\{1, \ldots, n\}$ and $\{1, \ldots, m\}$ so that X and Y may be assumed to belong to $\mathbb{Z}^n$ and $\mathbb{Z}^m$, respectively. Furthermore, E can be represented by the connectivity (or adjacency) matrix given by

$$E_{ij} = \begin{cases} 1, & \text{if } (i,j) \text{ is an edge} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. Next, let $G = (V, E, W)$ denote a *weighted bipartite graph* where (V, E) is a bipartite graph with $W \in \mathbb{R}^{n\times n}$ denoting the weight matrix associated with the edges. We assume that $W_{ij} \geq 0$, if $E_{ij} = 1$ and $W_{ij} = \infty$ if $E_{ij} = 0$.

A vertex $i \in X$ (resp., $j \in Y$) is *incident* to an edge if there exists $j \in Y$ (resp., $i \in X$) such that (i, j) is an edge. In this case, (i, j) are *adjacent*. A set of vertices $U \subseteq V$ is a *vertex cover* of G if the set $\{(i, j) : E_{ij} = 1, i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}\} \subseteq U$ that is, if every edge in E is incident with a vertex in U. A *minimum vertex cover* is a vertex cover of G with the minimum number of vertices. The existence of such a minimum vertex cover is easy to establish.

Two edges of a graph are *independent* if they do not have any common incident vertices. A *matching* of G is a set $M \subseteq E$ which is a *subgraph* of independent edges in a graph $G = (V, E)$. Since G is bipartite it follows that the set of incident vertices U of a matching will be of the form $\hat{X} \times \hat{Y}$ such that $\hat{X}$ and $\hat{Y}$ have the same number of vertices. A *maximum matching* of G is a matching of G that has the largest number of edges (and hence the largest number of incident vertices). A *perfect matching* of G is a (maximum) matching of G such that the set of incident vertices is $X \times Y$ (see Figure 2).

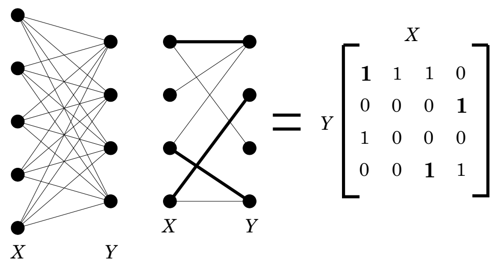

Fig. 2. A complete bipartite graph, a matching on a bipartite graph and the equivalent edge matrix

By definition, if a graph G has a perfect matching M then $m = n$, that is X and Y have the same number of vertices. Furthermore, $Me = M^T e = e$ where $e = \begin{bmatrix} 1 & 1 & 1 & \ldots \end{bmatrix}^T$. Note that these conditions are only necessary but not sufficient for the existence of a perfect matching. Hence, it follows that if a bipartite graph $G = (V, E)$ has a perfect matching then $Ee \geq\geq e$ and $E^T e \geq\geq e$. However, note that these conditions are not sufficient (for example, consider the graph given in Figure 3). The following result provides a necessary and sufficient condition for the existence of a perfect matching. First, however, we introduce the following notation. Let $\mathcal{N}_D : \mathbb{R}^{m\times n} \to \mathbb{R}$ be defined by

$$\mathcal{N}_D(E) \triangleq e_m^T D e_n \tag{2}$$

where $D \in \mathbb{R}^{m\times n}$ is given by

$$D_{ij} = \begin{cases} E_{ii}, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

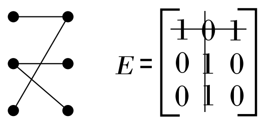

Fig. 3. A minimum line covering on an edge matrix

Note that if $m = n$ then $\mathcal{N}_D(E)$ denotes the number of one on the diagonal. Next, let $\nu_{max} : \mathbb{R}^{m\times n} \to \mathbb{R}$ be defined by

$$\nu_{max}(E) \triangleq \max_{P_m \in \Pi_m, P_n \in \Pi_n} \mathcal{N}_D(P_m E P_n) \tag{4}$$

where $\Pi_m \subseteq \mathbb{R}^{m\times m}$ and $\Pi_n \subseteq \mathbb{R}^{n\times n}$ denote the set of $m \times m$ and $n \times n$ *permutation matrices* [**?**], respectively.

Theorem 2.1: Consider a bipartite graph $G = (V, E)$ where $E \in \mathbb{R}^{m\times n}$. The size of the maximum matching is $\nu_{max}(E)$. Furthermore, if $m = n$ then G has a perfect matching if and only if $\nu_{max} = n$.

Proof: Let M denote a maximum matching of G. In this case it is easy to show that there exists permutations of X and Y such that the edge matrix M_Π of the permutations X_Π and Y_Π is such that $M_{\Pi_{ij}} = 0$, $i \neq j$ and the size of the matching M_Π is given by $e_m^T M_\Pi e_n = e_m^T M e_n$. The result now follows from the fact that there exist $P_m \in \Pi_m$ and $P_n \in \Pi_n$ such that $M_\Pi = P_m M P_n$ and $\mathcal{N}_D(P_m E P_n) = e_m^T M_\Pi e_n$.

Next, if $m = n$ it follows that G has a perfect matching if and only if $\nu_{max}(E) = n$ by noting that a perfect matching is a maximum matching of size n. ■

Remark 2.1: It follows from Königs theorem [13] that $\nu_{max}(E)$ is the size of the minimum vertex cover. Hence, $\nu_{max}(E)$ denotes the minimum number of lines (drawn across rows and columns) that are sufficient to cover all the 1's in E which implies that $\nu_{max}(E)$ is the maximum number of 1's no two of which are in the same line (row or column) of E

Remark 2.2: Although Theorem 2.1 gives necessary and sufficient conditions for the existence of a perfect matching it is in general more computationally efficient to use Königs theorem to compute $\nu_{max}(E)$.

Next, we consider the *virtual perfect matching problem* where given a graph $G = (V, E)$ that does not have a perfect matching we provide a method for constructing the *virtual graph* $\hat{G} = (\hat{V}, \hat{E})$ such that $\hat{G}$ has a perfect matching and G is a *projection* of $\hat{G}$, that is, G is a subgraph of $\hat{G}$ such that when all the vertices in $\hat{V}\backslash V$ are eliminated (along with their edges) then the resultant graph is G. Note that if $E \in \mathbb{R}^{m\times n}$ then by choosing $\hat{V} = \hat{X}\times\hat{Y} \subseteq \mathbb{R}^{m\times n}\times\mathbb{R}^{m\times n}$ such that all the n additional (virtual) vertices of $\hat{X}$ and the m additional virtual vertices of $\hat{Y}$ are connected with every other vertex of $\hat{Y}$ and $\hat{X}$, respectively, it can be shown that $\hat{G}$ has a perfect matching. The following result provides a construction $\hat{G}$ that has the minimum number of virtual vertices.

Theorem 2.2: Let $G = (V, E)$ be a bipartite graph. Then there exists $\hat{G} = (\hat{V}, \hat{E})$ such that G is a projection of $\hat{G}$, $\hat{G}$ has a perfect matching, and

$$\nu_{max}(\hat{E}) = (m + n) - \nu_{max}(E) \tag{5}$$

Furthermore, there does not exist $\tilde{G} = (\tilde{V}, \tilde{E})$ such that G is a projection of $\tilde{G}$, $\tilde{G}$ has a perfect matching, and $\nu_{max}(\tilde{E}) < \nu_{max}(\hat{E})$.

Proof: Let M be a maximum matching and assume without loss of generality that $M_{ij} = 0$, $i \neq j$. Now, it can be shown that by adding $n - \nu_{max}(E)$ and $m - \nu_max(E)$ virtual vertices to X and Y the resultant $\hat{G}$ has a perfect matching. The minimality of $\nu_{max}(\hat{E})$ given by (5) can be shown in a similar manner. ■

Remark 2.3: Note that if G has a perfect matching, that is, $\nu_{max}(E) = m = n$ is follows that $\nu_{max}(\hat{E}) = \nu_{max}(E)$ and hence $G = \hat{G}$ (since G is a projection of $\hat{G}$)

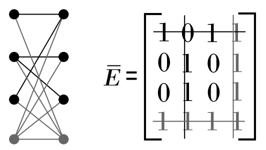

Fig. 4. Augmented edge matrix with the new minimum line cover

Finally, we consider the *optimal assignement problem*. Specifically, let $G = (V, E, W)$ be a weighted bipartite graph that has at least on perfect matching. The optimal assignment problem is the problem of finding a perfect matching that has the *minimum weight*, that is, finding $\overline{M}$ such that

$$\overline{M} = \min_{M\in\mathcal{M}} e^T(\mathcal{M} \circ M)e \tag{6}$$

where $\mathcal{M}$ denotes the set of all perfect matchings of G. More generally, let $G = (V, E, W)$ be a weighted bipartite graph such that $E \in \mathbb{R}^{m\times n}$ which may not have a perfect matching. In this case, the optimal assignment problem is given by

$$\overline{M} = \min_{\hat{M}\in\hat{\mathcal{M}}} e^T(\hat{W} \circ \hat{M})e \tag{7}$$

where $\hat{M}$ denotes the set of perfect matchings of $\hat{G}$, the virtual graph of G, and $\hat{W}$ denotes the weighting matrix of $\hat{G}$ such that $\hat{W}_{ij} = 0$ if i or j correspondes to a virtual vertex and $\hat{W}_{ij} = W_{ij}$ otherwise.

Note that the construction of $\hat{G}$ can be performed using the algorithm described in [16] and the optimal assignment problem given by (7) can be solved by using the Hungarian method [5], [13], [16].

III. Decentralized Decision Algorithm

The Hungarian Method was meant to solve for a minimum weight perfect matching on a static graph, but a more relevant problem with regards to the application of interest in this paper is finding a minimum weight perfect matching as a graph is dynamically changing. In this section we will define the problem of minimizing the total edge weight for a dynamic graph. We will also discuss the conditions for applying this algorithm to a dynamic graph.

A *dynamic graph* is a graph where the vertices, vertex weights, edge matrix, and edge weight are changing with time. Since we are dealing with a matching on a bipartite graph, vertex weights are not used and can be assumed constant. We will denote a dynamic graph by $\mathcal{G}(t) = (\mathcal{V}(t), \mathcal{E}(t), \mathcal{W}(t))$ where $\mathcal{E}(t)$ and $\mathcal{W}(t)$ are $n\times n$ matrices.

In the static case, the edge weight matrix is a matrix of positive constants. In the dynamic case $\mathcal{W}$ depends on time and the time-varying edge matrix. This leads to the general definition of a time-varying weight matrix for a bipartite graph given by

$$\mathcal{W}_{ij}(t, \mathcal{E}(t)) \triangleq \begin{cases} f_{ij}(t) & \mathcal{E}_{ij}(t) = 1 \\ \infty & \mathcal{E}_{ij}(t) = 0 \end{cases} \tag{8}$$

where $f_{ij}(t) : \mathbb{R} \to \mathcal{I} \subset \mathbb{R}_+$ are a set of functions that describe the weight function for a particular edge of $\mathcal{G}$ and are not necessarily smooth or continuous, and $\mathcal{I}$ is a closed bounded interval.

The choice of the weight function is determined by the problem objective. In the case of LEO sensing or imaging constellations, the objective is to maximize the time that the satellites are tracking a target. This is equivalent to minimizing the time a the satellites spend moving between targets. For this problem the optimal assignment problem is to minimize the integral of the total weight of the matching over an interval of time.

The total weight of the matching is given by

$$w(\mathcal{M}) \triangleq \sum_{i=1}^{n}\sum_{j=1}^{n} \mathcal{W}_{ij}(t, \mathcal{E}(t))\mathcal{M}_{ij} \tag{9}$$

This leads to the continuous optimal assignment problem given by

$$\overline{\mathcal{M}}(t) = \int_{t_1}^{t_f} \min_{\mathcal{M}} w(\mathcal{M})\mathrm{dt} \tag{10}$$

$$= \int_{t_1}^{t_f} \min_{\mathcal{M}} \sum_{i=1}^{n}\sum_{j=1}^{n} \mathcal{W}_{ij}(t)\mathcal{M}_{ij} \tag{11}$$

where $\mathcal{M}(\mathcal{E}, \mathcal{W})$ is the set of admissible matchings at time t.

Equation 9 shows that the behavior of the total weight function is dependent on the matching and the edge weight functions. For example, if the matching does not change on the time interval and the edge weight functions are continuously differentiable, then the total weight function will be continuously differentiable on the interval. On the other hand, if the edge weight function is constant on the time interval, the matching will only change when the edge matrix changes. This means that the matching and edge weight functions must be "well behaved" for the integral to behave nicely.

In the following we show that if the total edge weight is bounded on the time interval, then the solution is optimal and the finite time version of the problem approaches the optimal solution in the limit. This is equivalent to a perfect matching existing and $f_{ij}(t)$, $i, j = 1, 2, \ldots, n$ bounded on $[t_1, t_f]$.

In actual implementation, the algorithm will be applied in a discrete manner with the matching changing only at discrete times. In the following we show that as the time between the matching updates decreases, the algorithms behavior approaches the continuous time solution.

We will start by defining a discrete time version of the optimal assignment problem. We will allow the matching to be made at discrete times in the time interval. Let $P = \{t_k\}_{k=1}^{f}$ be a sequence of times that partition the interval (t_1, t_f). Now let

$$w^k(\mathcal{M}) \triangleq \sum_{i=1}^{n}\sum_{j=1}^{n} \mathcal{W}_{ij}(t_k)\mathcal{M}_{ij} \tag{12}$$

be the total edge weight of a matching at time t_k where $\mathcal{M}$ are admissible matchings. The total edge weight $w^k(\mathcal{M})$ is constant on the interval $[t_k, t_{k+1})$. The minimum weight matching is calculated at the beginning of each time interval and remains constant over the interval and is given by

$$\overline{\mathcal{M}}(t_k) = \min_{\mathcal{M}} w^k(\mathcal{M}) \triangleq \min_{\mathcal{M}} \sum_{i=1}^{n}\sum_{j=1}^{n} \mathcal{W}_{ij}(t_k)\mathcal{M}_{ij} \tag{13}$$

The total weight of the matching over the interval is the weight at the beginning of each subinterval time the length of the interval, and the optimization problem becomes

$$\sum_{k=0}^{f-1} \min_{\mathcal{M}} w^k(\mathcal{M})(t_{k+1} - t_k) \tag{14}$$

$$= \sum_{k=0}^{f-1} \min_{\mathcal{M}} \sum_{i=1}^{n}\sum_{j=1}^{n} \mathcal{W}_{ij}(t_k)\mathcal{M}_{ij}(t_{k+1} - t_k) \tag{15}$$

$$= \sum_{k=0}^{f-1} \overline{\mathcal{M}}(t_k)(t_{k+1} - t_k) \tag{16}$$

$$\tag{17}$$

Next, to show that as the partition gets finer, the discrete-time problem approaches the continuous time problem, we must show that the total edge weight is Riemann integrable. Let

$$g(t_k) = \min_{\mathcal{M}} \sum_{i=1}^{n}\sum_{j=1}^{n} \mathcal{W}_{ij}(t_k)\mathcal{M}_{ij} = \overline{\mathcal{M}}(t_k) \tag{18}$$

and let

$$I(g) = \lim_{\|P\|\to 0} \sum_{k=1}^{f} g(t_k)(t_k - t_{k-1}) \tag{19}$$

$$= \lim_{\|P\|\to 0} \sum_{k=1}^{f} \overline{\mathcal{M}}(t_k)(t_k - t_{k-1}) \tag{20}$$

where $\|P\| = \max_P |t_k - t_{k-1}|$. Since $\overline{\mathcal{M}}(t_k)$ is constant and finite on the interval $[t_k, t_k + 1)$, $I(g)$ exists and is finite. Hence, by Riemann integrability

$$\overline{\mathcal{M}}(t) = \int_{t_1}^{t_f} \min_{\mathcal{M}} w(\mathcal{M})\mathrm{dt} \tag{21}$$

$$= \lim_{\|P\|\to 0} \sum_{k=1}^{f} \overline{\mathcal{M}}(t_k)(t_k - t_{k-1}) \tag{22}$$

which shows that in the limit, the discrete time case approaches the continuous time optimal solution.

IV. Satellite Application

In the following we will provide additional practical details to applying the Hungarian method to the problem of interest. Applying this algorithm to a dynamic system involves selecting the weight functions based on the dynamics of the system that capture the behavior to be optimized. The dynamic system of interest in this paper is a constellation of low-earth orbit (LEO) satellites. The function of these satellites is to track a specific location on the earth for communications, imaging, etc.

We consider, the problem where the targets are not necessarily known and changing with time. Another dynamic aspect of this problem, that arises because of the low-earth orbit, is that targets are constantly moving in and out of range of a satellite changing the edges of the graph.

We start by defining the satellite target system as a weighted bipartite graph. The set of all available targets will be denoted by $\overline{\mathcal{T}}(t)$ and the set of all satellites will be denoted by $\overline{\mathcal{S}}(t)$. Due to the dynamics of the problem each satellite will only have a subset of the targets in range at any time. Also, at any given time, there may be satellites with no targets in range and targets that are not in range of any satellite. These targets and satellites cannot be matched so they have no bearing on the solution and just add to the computational load, so we define the subset of targets $\mathcal{T}(t) \subseteq \overline{\mathcal{T}}(t)$ as the instantaneous set of targets $\{T_1, \ldots, T_m\}$ that are in range of a satellite, similarly we define $\mathcal{S}(t) \subseteq \overline{\mathcal{S}}(t)$ as the instantaneous set of satellites $\{S_1, \ldots, S_m\}$ with at least one target in range.

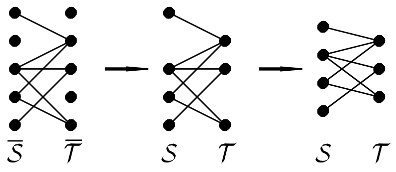

Fig. 5. Removing the targets and satellites with no edges incident

We will assume that there are $i = 1, \ldots, n(t)$ satellites and $j = 1, \ldots, m(t)$ targets in $\mathcal{S}(t)$ and $\mathcal{T}(t)$ at time $t \in [t_1, t_f]$.

Next we want to describe the targets in $\mathcal{T}$ that are in range of a specific satellite in $\mathcal{S}$ at any time. We define $\mathcal{T}^i(t) \subseteq \mathcal{T}$ as the set of targets that are in range of the i^{th} satellite. Using Fig 5 as an example $\mathcal{T}^1 = \{T_1\}$, $\mathcal{T}^2 = \{T_1, T_2, T_3\}$, $\mathcal{T}^3 = \{T_1, T_3\}$, and $\mathcal{T}^4 = \{T_2\}$.

Note that

$$\bigcup_{j=1,\ldots,n} \mathcal{T}^j(t) = \mathcal{T}(t), \quad t \in [t_0, t_f]$$

The definition of $\mathcal{T}^i(t)$ allows us to define the edge matrix

$$E_{ij}(t) = \begin{cases} 0 & T_j \notin \mathcal{T}^i(t) \\ 1 & T_j \in \mathcal{T}^i(t) \end{cases} \tag{23}$$

where $i = 1, \ldots, n(t)$, $j = 1, \ldots, m(t)$. For Fig. 5 the edge matrix is

$$E = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{24}$$

The edge matrix defines the set of valid matchings at that instant in time. We will define the weight matrix by

$$W_{ij}(t, E(t)) \triangleq \begin{cases} f_{ij}(t) & E_{ij} = 1 \\ \infty & E_{ij} = 0 \end{cases} \tag{25}$$

where $W_{ij} : [t_1, t_f] \times \{0, 1\} \rightarrow \mathbb{R}_+$. For this satellite system, we want to maximize the time that the satellites spend looking at a target or minimize the time that the satellite spends changing its attitude to track a target. The time that a satellite takes to make an attitude maneuver is proportional to the angle that the satellite must slew through which we will denote by $\theta(t)$.

For each target in range of a satellite there is a slew angle to look at that target based on the target location and the satellite's current attitude. We will denote this angle by $\theta_{ij}(t)$. Because the range of the weight assignment functions is the positive real space we define the functions by

$$f_{ij}(t) = (\theta_{ij}(t))^2 \tag{26}$$

For a given weight matrix the total weight of a matching at time $t \in [t_1, t_f]$ given by

$$w(M) = \sum_{i=1}^{n} \sum_{j=1}^{m} (\theta_{ij}(t))^2 M_{ij} \tag{27}$$

is proportional to the sum of the total angle that each satellite must slew to track a given matching of targets. This means that the optimal assignment perfect matching with yield the matching that minimizes the sum of the slew angles that the satellites must move through to track those targets.

Assuming that the satellite has an asymptotically stable attitude controller, this means that for a constant matching the total matching weight is going asymptotically to zero. The speed with which the satellite can acquire a target can effect the algorithm adversely. If the attitude control system is slow, it may take longer to slew to track the target, then the time the target is in range. Figure 6 shows a simulation of how the total weight of the matching varies with time.

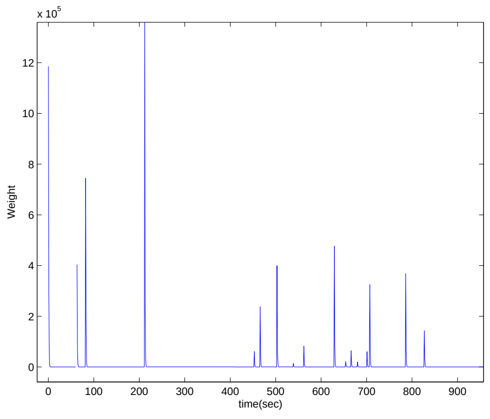

Fig. 6. The total weight of the matching versus time

V. Simulation Setup

In the following we will discuss the simulation of the algorithm on various LEO satellite constellations used for

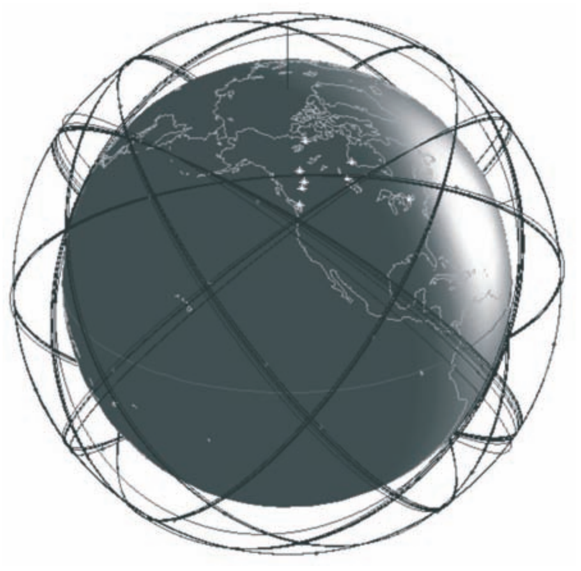

Fig. 7. GlobalStar satellite consellation screenshot from the simulation

sensing, and the simplifying assumptions used in the simulation. The simulation was used to validate the algorithm with satellite attitude and orbital dynamics added. The simulation was also used to look for computational issues with the algorithm and test situation that might lead to the algorithm becoming "unstable" for example, dithering between two targets without tracking either. Using realistic parameters and orbits is given less emphasis than providing a scenario where the behavior of the graph is rich and complex enough to thoroughly test the algorithm. To achieve this, some of the test orbits were chosen to have a group of satellites converging on a region of the earth with multiple targets. Many of these orbits would be impractical to launch a satellite into, but all orbits are feasible. The algorithm was also tested on various global coverage satellite constellations. The orbital propagation uses standard Keplarian two body dynamics and accounts for the earth's oblateness but not J2 effects.

For simplicity, the satellites simulated are assumed to be axisymmetric where the dynamics are given by

$$\dot{x}(t) = I \begin{bmatrix} \theta \\ \phi \\ \psi \end{bmatrix} + u(t) = Ix(t) + u(t), \quad I \in \mathbb{R}^{3\times 3} \tag{28}$$

where θ, ϕ, and ψ are Euler angles. The satellite is assumed to have a maximum slew angle giving the satellite a clearly defined footprint where it can track targets. This is done to prevent the satellite from looking at a target with a low angle of incidence.

A proportional integral attitude controller was chosen with gains that provide an average slew time of 10 seconds. When the satellite does not have a target to track, it tracks its ground track to return the attitude to the center of the footprint.

The simulation was also used to test the robustness of the algorithm to communication failures. The algorithm was implemented in a distributed fashion which improves the robustness of the algorithm. In the event of a communication failure, the behavior of the algorithm becomes suboptimal, but the overall objective of the satellite constellation is still accomplished. In all cases tested the algorithm performed as planned.

VI. Conclusion

In this paper we investigate the optimal assignment of targets to LEO satellites with narrow beamwidths. We extend the optimal assignment problem to dynamic weighted bipartite graphs and show the conditions on the edge weight assignment functions that lead to the optimal solution of the assignment problem for dynamic graphs. We also provide conditions for the existence of a perfect matching on a given bipartite graph, and a method for constructing the smallest virtual graph that does not change the minimum matching on the original graph and guarantees the existence of a perfect matching. Finally we apply the algorithm to a constellation of LEO satellites and verify by simulation the effectiveness of the algorithm.

Future extensions to this work include investigating the effects of time delay on the optimality of the algorithm and investigating the behavior and uses of different edge weight functions.

References

[1] M. Mesbahi, "On a dynamic extension of the theory of graphs," in *Proc. Amer. Contr. Conf.*, Anchorage, AK, 2002, pp. 1234–1239.

[2] L. P. F. Giulietti and M. Innocenti, "Autonomous formation flight," *IEEE Control Systems Magazine*, vol. December, pp. 34–44, 2000.

[3] R. M. M. D. P. Spanos, "Motion planning with wireless network constraints," in *Proc. Amer. Contr. Conf.*, Portland, OR, 2005, pp. 87–92.

[4] D. S. N. Gordon, "Aspects of target tracking: Problems and techniques," in *IEEE Collq. on Target Tracking and Data Fusion*, Birmingham, UK, 1998, pp. 1–6.

[5] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Res. Logist. Quarterly*, vol. 2, pp. 83–97, 1955.

[6] H. N. Gabow and R. E. Tarjan, "Faster scaling algorithms for newtork problems," *SIAM J. Comput.*, vol. 18, pp. 1013–1036, 1989.

[7] W. K. S. M. Y. Kao, T. W. Lam and H. F. Ting, "A decomposition theorem for maximum weight bipartite matchings," *SIAM J. Comput.*, vol. 31, pp. 18–26, 2001.

[8] F. Harary and G. Gupta, "Dynamic graph models," *Mathl. Comput. Modelling*, vol. 25, pp. 79–87, 1997.

[9] M. Mesbahi, "State-dependent graphs," in *Proc. IEEE Conf. Dec. Contr.*, Maui, HI, 2003, pp. 3058–3063.

[10] E. Klavins and R. M. Murray, "Distributed algorthms for cooperative control," *IEEE Pervasive Computing*, vol. 3, pp. 56–65, 2004.

[11] A. Y. Zomaya, "Scheduling: Theory and applications," *Int. Journ. of Foundations of Comp. Sci.*, vol. 12, pp. 559–564, 2001.

[12] C. S. L. C. L. Chen and C. D. McGillem, "Task assignment and load balancing of autonomous vehicles in a flexible manufacturing system," *IEEE Journ. of Robotics and Automation*, vol. RA-3, pp. 659–671, 1987.

[13] K. Thulasiraman and M. N. S. Swamy, *Graphs: Theory and Algorithms*. New York: Wiley, 1992.

[14]

[15] J. McHugh, *Algorithmic Graph Theory*. New Jersy: Prentice Hall, 1990.

[16] G. V. Shenoy, *Linear Programming: Methods and Applications*. New York: Wiley, 1989.