Pittsburgh

SAND 97-2014C
CONF-97-2014C
CONF-971086--

Analyzing the Multiple-target-multiple-agent Scenario Using Optimal Assignment Algorithms

Kwan S. Kwok

Brian J. Driessen

Cynthia A. Phillips

(Sandia National Laboratories, Albuquerque, NM 87185-1003)

Craig A. Tovey

(Georgia Institute of Technology, Atlanta, GA 30332)

RECEIVED

SEP 23 1997

OSTI

ABSTRACT

This work considers the problem of maximum utilization of a set of mobile robots with limited sensor-range capabilities and limited travel distances. The robots are initially in random positions. A set of robots properly guards or covers a region if every point within the region is within the effective sensor range of at least one vehicle. We wish to move the vehicles into surveillance positions so as to guard or cover a region, while minimizing the maximum distance traveled by any vehicle. This problem can be formulated as an assignment problem, in which we must optimally decide which robot to assign to which slot of a desired matrix of grid points. The cost function is the maximum distance traveled by any robot. Assignment problems can be solved very efficiently. Solution times for one hundred robots took only seconds on a Silicon Graphics Crimson workstation. The initial positions of all the robots can be sampled by a central base station and their newly assigned positions communicated back to the robots. Alternatively, the robots can establish their own coordinate system with the origin fixed at one of the robots and orientation determined by the compass bearing of another robot relative to this robot.

This paper presents example solutions to the multiple-target-multiple-agent scenario using a matching algorithm. Two separate cases with one hundred agents in each were analyzed using this method. We have found these mobile robot problems to be a very interesting application of network optimization methods, and we expect this to be a fruitful area for future research.

Keywords: cooperative behavior, multiple agent, emergent behavior, network optimization, multiple target, matching, graphs

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Further author information-

K.S.K.: Email: kskwok@sandia.gov; Telephone: 505-845-7170

B.J.D.: Email: bjdries@sandia.gov; Telephone: 505-284-2047

C.A.P.: Email: caphill@sandia.gov; Telephone: 505-845-7296

C.A.T.: Email: ctovey@isye.gatech.edu; Telephone: 404-894-3034

MASTER

DISCLAIMER

**Portions of this document may be illegible
in electronic image products. Images are
produced from the best available original
document.**

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

1. Introduction

There has been significant recent interest in the use of multiple robotic agents or robotic vehicles to achieve tasks that a single robot could not achieve alone or that could be accomplished much more efficiently or robustly with multiple robotic devices. There are various approaches to controlling multiple robotic devices. One is to have each behave independently or autonomously, but receive information from its neighbors. The receipt of such information could be limited to neighbors that are within a certain communication range (for covertness) of the robotic device. The other extreme is to have a centralized base station, either mobile or fixed, that collects position and other information on all of the vehicles and commands them to move to certain locations.

Reynolds [19] considered the formation of flocks, herds, and schools in simulations in which multiple autonomous agents were driven away from each other and other obstacles by inverse square law repulsive forces. Part of the motivation behind Reynold's work was the impression of centralized control exhibited by actual bird flocks, animal herds, and fish schools, despite the fact that each agent (bird, animal, or fish) is responding only to its limited-range local perception of the world, in Reynold's opinion. Reynolds stated "Natural flocks seem to consist of two balanced, opposing behaviors: a desire to stay close to the flock and a desire to avoid collisions within the flock." He also stated "A bird might be aware of three categories: itself, its two or three nearest neighbors, and the rest of the flock." The desire to be close to the flock while avoiding collisions was modeled by force fields that attracted a bird to its two to three closest neighbors but prevented it from getting too close to these neighbors.

Brooks [6] examined the problem of an autonomous agent or mobile robot with "multiple goals." For example, the robot may have goals of (1) moving North and (2) avoiding obstacles, where the second has higher priority than the first. Instead of the force field type of approach, Brooks had the robot take a mixture of control laws or states of a finite state machine. For example, one control law (or "finite state") may be associated with moving North and one associated with obstacle avoidance. The moving-North control law would be essentially replaced by an avoid-obstacle control law if the impending collision danger level became large enough, although both control laws could be acting simultaneously and independently, in his subsumption architecture. The concept of cooperation among robots or the ability of multiple robots to perform tasks more effectively was not specifically addressed. More discussion of the subsumption architecture can be found in Brooks and Flynn [5], in which there is some discussion of the advantages of having massive numbers of autonomous agents.

Arkin [4] studied an approach to "cooperation without communication," for multiple mobile robots. He states that "by enabling many simple robots to cooperate together on a large task, it becomes possible to solve a problem that would be infeasible

using a large-scale machine." Arkin also points out that centralized master/slave or hierarchy-based approaches have drawbacks of potential communication bottlenecks and less robustness than completely decentralized approaches in which each agent is autonomous and has the same control law. Arkin demonstrated such an approach for robots that are supposed to forage and retrieve objects (in a hostile environment). Anticipated applications mentioned were "housekeeping on the exterior of a space station or undersea base" and "mining operations at remote locations." Kube and Zhang [14] also considered decentralized robots performing tasks "without explicit communication." Much of their study examined comparisons of behaviors of different social insects such as ants and bees. They considered a box-pushing task and utilized a subsumption approach as well as ALN (Adaptive Logic Networks).

Noreils [17] conducted work toward achieving cooperation between mobile robots for indoor environment applications. He dealt with robots that were not necessarily homogeneous. That is, one subset of the robots may have capabilities that another subset does not have. His architecture consisted of three levels: functional level, control level, and planner level. The planner level was the high-level decision maker. He developed a special purpose language for his application.

Chen and Luh [7] examined decentralized control laws that drove a set of mobile robots into a specific formation. In particular, a circle formation was considered. Similarly, Yamaguchi and Arai [21] studied line-formations, and so did Yoshida et al, [22].

2. Problem Statement

The problem of immediate interest is as follows. We want many (up to 100 or more) disposable (small) vehicles to "stake out" a dangerous area. The vehicles are air-dropped and thus land in a fairly random set of positions. We either have a base station that receives actual positions (relative to the base station) of each vehicle and also sends commanded positions to each vehicle, or we may have one of the vehicles itself serve as a "base" vehicle. In the latter case, triangulation can be used to obtain the position of any vehicle relative to an imaginary reference frame fixed to, for example, vehicles number 1, 2, and 3. Vehicles 4 through 100 will then be commanded to move relative to this frame. It is assumed that each vehicle has a compass so that it can move in the correct direction to obtain a new desired location within this frame, and it is assumed that the orientation of this frame is determined by the fact that vehicle 1 knows the relative bearing of vehicle 2.

Each vehicle is equipped with a sensor for detecting the subject(s) (or "targets") of the search. These detection sensors have limited range. Yet, we want to be sure that no

subject (or "target") is presently amidst or becomes amidst the group without being detected. Thus, there cannot exist regions or gaps between vehicles inside which a target could "hide" or go undetected. In general, to fill in an area, without any gaps, the vehicles are going to have to move, as they are initially in essentially random positions. Yet, each vehicle has a limited amount of fuel/power with which to locomote itself and subsequently observe. Therefore, minimizing the distance that the vehicles travel to adequately "stake out" the area is critical since it determines how long the surveillance grid will be "up" before at least one vehicle loses power. It is assumed that the spacing between the vehicles is large relative to the size of the vehicles, so that collision between moving vehicles is not a real issue.

3. Solution Method

The approach used herein to tackle this problem is to create an imaginary set of equally-spaced grid points in the reference frame, forming essentially a square, or nearly square, area. The spacing between the grid points is less than the maximum target-detection range of each vehicle's detection sensor. This guarantees that any target within this rectangular area will be detected. The number of grid points is equal to the number of vehicles, and we will assign each vehicle to one of these grid points, in such a way that the maximum distance any vehicle must travel is minimized. This is, in essence, an assignment problem (which vehicle to which grid point). As we will see, in the next section, assignment problems of this type can be solved *very* efficiently combinatorial algorithms, which will be described in the next section. By minimizing the maximum distance traveled by any vehicle, we maximize the potential time the region will be properly guarded. We can also determine if the specified region cannot be covered at all because some vehicle would have to move farther than its maximum travel range. The grid point locations are herein assumed to be determined a priori, as opposed to being part of the optimization variable set.

The case of minimizing the sum of the distances traveled by all bugs, subject to no bug moving more than its fuel resources allow was also considered. Numerical results for this case are provided as well in section 5.

4. Solution of the Vehicle Assignment Problem

In this section we discuss some of the methods of solving the assignment problem. In this section we will use the term "slot" for "grid point". Let d_{ij} be the distance from vehicle i to slot j .

We begin with some preliminary definitions. A *graph* $G = (V, E)$ is a set of *nodes* V and a set of *edges* connecting pairs of nodes. We will assume an *undirected* graph, where edges are unordered pairs (v_i, v_j) for $v_i, v_j \in V$. We say that edge (v_i, v_j) is *adjacent* to nodes v_i and v_j . We can also assign *weights* to the edges. A *matching* in a graph is a set of edges such that each node is adjacent to at most one edge in the set. Thus a matching represents a pairing of the nodes. In a *perfect* matching, each node is adjacent to exactly one edge (each node is paired with another). A *maximum-cardinality* matching, is a matching with the maximum possible number of edges*. A *minimum weighted* matching is a maximum-cardinality matching for which the sum of the weights of the selected edges is as small as possible. In a *bipartite* graph, the node set can be partitioned into two sets A and B such that all the edges go between a node in the A set and a node in the B set. The classic *assignment problem* is finding a minimum- (or maximum-) weight matching in a bipartite graph.

We can formulate the two versions of the robot-assignment problem as matching problems in bipartite graphs. To assign the robots to slots so as to minimize the maximum distance traveled, we guess the maximum distance D , say half the maximum d_{ij} . We then form the following bipartite graph G_D : Node set A has a node r_i for robot i . Node set B has a node s_j for each slot j . The edge (r_i, s_j) is in set E if and only if $d_{ij} \leq D$. We then compute a maximum cardinality (i.e. unweighted) matching in this graph. There is a legal assignment of robots to slots (i.e. one which covers all the slots and has no robot traveling more than distance D), if and only if there is a perfect matching in this graph. We can perform a binary search on D to find the smallest legal value of D , and hence minimize the maximum distance traveled by any robot.

To minimize the average distance traveled by all robots while obeying a maximum distance bound D , we form graph G_D as above, but now each edge (r_i, s_j) is given weight d_{ij} . We then compute a minimum-weight matching on this graph. If there is no perfect matching, then the distance bound D is too small.

Frequently in practice, network optimization problems such as network flow, min-cost flow, and the transportation problem are solved using the *network simplex* algorithm. This involves formulating the problem as a linear program (LP). For example the LP for the assignment problem is:

$$\text{Min } \sum_i \sum_j d_{ij} x_{ij}$$

subject to

$$\sum_i x_{ij} = 1 \quad \forall j$$

$$\sum_j x_{ij} = 1 \quad \forall i$$

* Note that a graph need not have a perfect matching

$$x_{ij} \geq 0 \quad \forall i, j$$

(where in the LP's solution, x_{ij} will be 1 if and only if vehicle i will go to slot j , and 0 otherwise.) What we really require is an *integer programming* formulation where the x_{ij} must take on the values 0 or 1 only. However, the matrix for linear programs with network structure is totally *unimodular* and therefore every extreme point of the polytope of feasible solutions has integer variable values. Therefore the optimal solution to the LP will in fact give the optimal solution for the integer program. One can solve linear programs using the simplex method, which starts at a corner (extreme point) and pivots from corner to corner (while the objective function improves monotonically), until it reaches an optimal solution. Hillclimbing works in this setting since the feasible region is convex. In general, the simplex method will not terminate in polynomial time in the worst case. However, for problems with network structure, specialized pivoting rules can lead to polynomial-time algorithms. Some of the most recent examples of polynomial-time specializations to the assignment problem include [1, 3, 20]. See [18] for other examples.

Typically the worst-case complexity of network simplex, even for the polynomial-time variants, is worse than combinatorial (graph-based) algorithms. However, they frequently perform better in practice because they are so simple. For the assignment problem, however, the LP formulation is highly *degenerate*, meaning there are many ways to represent each corner [13]. As a consequence, the network simplex algorithm typically makes many pivots which leave the objective function unchanged. Therefore, for the assignment problem and related matching problems, graph-based algorithms are strong competitors in practice.

The classic combinatorial algorithm for finding a maximum (weighted or unweighted) matching in a graph is due to Edmonds [9]. We will begin with the unweighted case. The method works by repeatedly finding *augmenting paths*. Suppose we have a graph $G = (V, E)$ and some matching $M \subset E$, which is not necessarily of maximum size. An augmenting path begins and ends at a *free node*, namely one which is not adjacent to any edge in the matching. Obviously the first edge and last edge are not in the matching. The edges along the path alternate between those in the matching and those not in the matching. Thus the path has an odd number of edges, say $2q+1$, of which $q+1$ are not in the matching, and q are. More formally, an augmenting path is a path in the graph $v_0, v_1, \dots, v_k$, where k is odd, no edge in M is adjacent to either v_0 or v_k , and for each edge (v_i, v_{i+1}) in the path, the edge is in matching M if i is odd, and the edge is not in the matching if i is even. If we consider only edges on this path, and reverse their roles (remove all those previously in the matching and replace them by the ones that were not), then we still have a valid matching, and it has one more edge than it did before the augmentation. A matching is maximum if and only if there are no more augmenting paths in the graph. For the (maximum) weighted matching problem, an augmenting path

is an alternating path where the sum of the weights of the edges not in the matching is greater than the sum of the weights of the edges in the matching.

Many of the advances in matching algorithms have come from finding better implementations of this basic algorithm. The theoretically best algorithm for unweighted matching in general graphs is due to Micali and Vazirani [16] and runs in time $O(e\sqrt{v})$ where e is the number of edges in the graph and v is the number of nodes. There are no theoretically better algorithms for bipartite graphs, though in practice the implementation of the search for augmenting paths is simpler. The theoretically best algorithm of which we are aware for weighted matching in general graphs is due to Gabow [10] which runs in time $O(v(e + v \log v))$, and again we are aware of no specializations to bipartite graphs which are theoretically superior. There are also algorithms based on *scaling* the weights of the edges. They can be superior theoretically when the weights are small relative to the number of nodes in the graph. For example [2] quotes a bound of $(\sqrt{v} \log(vC))$, where C is the maximum weight. They give an overview of cost-scaling algorithms for the assignment problem and the popular Hungarian algorithm. For a much more detailed discussion of the Hungarian algorithm, see [13].

The code actually used to solve the matching problems for the vehicle assignment problem we describe here was acquired from the center for Discrete Mathematics and Theoretical Computer Science (DIMACS). These were codes made available from the first DIMACS implementation challenge for network flows and matching (1990-1991). The codes are available by following pointers to the implementation challenge off the DIMACS home page <http://www.dimacs.rutgers.edu>. The maximum-cardinality matching code is Ed Rothberg's implementation of Gabow's $O(v^3)$ implementation of Edmonds' algorithm [11]. The weighted matching code is Rothberg's implementation of Gabow's $O(v^3)$ implementation of Edmonds' weighted matching algorithm [12]. Note that these codes find matchings for *general* graphs, not just bipartite graphs, and that by using algorithms specialized to bipartite graphs, we could probably solve our problems even faster.

5. Example Vehicle Assignment Problem Solutions

Suppose that there are 100 vehicles. In Figure 1, the large dots represent grid points or "slots" to be filled. The spacing between these slots is less than the maximum sensor detection range of each vehicle. The initial random positions of the vehicles are shown in Figure 1 with small dots. The connecting lines in Figure 1 show the path traveled by each vehicle in moving to its assigned slot. The spacing between the slots is 0.1 units, and the maximum distance traveled by any vehicle was 0.3 units. The computer platform on which the numerical cases were run was a Silicon Graphics Crimson, running IRIX 5.2. The C code was compiled with the cc compiler, using no

optimization options. The problem considered in Figure 1 was to minimize the maximum distance traveled by any vehicle. The computer run time was 21 seconds. The resulting maximum distance traveled was 0.3 units, and the total distance traveled (summed over all vehicles) was 26 units.

Figure 1. Graphical Illustration of Results for a Min-Max Distance Traveled Problem

We also solved the problem of minimizing the total distance traveled subject to no vehicle moving more than 0.5 units. The computer run time on the Silicon Graphics Crimson was 3.6 seconds. The resulting solution is depicted in Figure 2 below.

Figure 2. Graphical Illustration of Results for a Minimize the Total Distance Traveled Subject to Fuel Constraints Problem

In contrast to the min-max solution, the maximum distance traveled by any vehicle was 0.49 units (compared to 0.3 units for the min-max solution). The total distance traveled, however, was smaller, namely 15.9 units, as compared to 26 units for the min-max solution.

6. Other Vehicle-Related Network Optimization Problems

In this paper we assumed that the grid was given. One could consider also optimizing the grid-point locations. For example, given a specific grid size, one could choose a grid-center location for which the robotic vehicle assignment problem required the smallest maximum distance travelled. Alternatively, one could trade-off size of the region covered versus redundancy of point coverage, and also consider the ability of robots to travel to the points.

Another problem of current interest is that of mine detection in specified regions. Clearly, 100 robotic vehicles equipped with mine-detection sensors could more quickly scan an entire specified region for mines than could an equally equipped lone vehicle. In this problem, again the sensor detection range is limited. That is, a vehicle must be within a certain distance of a mine to be able to detect it. One possible approach to this problem is to fill the specified area with imaginary grid points whose spacing is less than the maximum sensor range. Then, to most efficiently (or most quickly) scan the area, we would require that each grid point be visited by at least one vehicle in the least amount of time. The solution to this problem would be a time trajectory of each vehicle. This problem is an instance of the classic vehicle routing problem, which is a variant of the Traveling Salesman Problem. Special solution methods are currently being investigated.

7. Conclusion

This work used matching algorithms to optimally utilize the limited-range sensor capability and limited-travel capability of a group of robotic vehicles that were tasked to "stake out" an area. The solution method was to create an array of equally-spaced grid points, whose number is the same as the number of vehicles, and with the distance between the grid points less than the maximum sensor detection range of each vehicle. Then, the problem of minimizing the maximum distance traveled by any vehicle in the course of placing one vehicle at each grid point becomes an assignment problem (which vehicle to which grid point). There exist readily available algorithms for efficiently solving such assignment problems. The network simplex method was discussed. Matching algorithms were also discussed for solving the same assignment problems, and two example problems solved with this method.

Acknowledgments

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

References

- [1] Ahuja, R. and Orlin, J., "The Scaling Network Simplex Algorithms," *Operations Research*, Vol. 40, Supplement 1, 1992, pp. S5-S13.
- [2] Ahuja, R., Magnanti, J. and Orlin, J., *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, N.J., 1993.

- [3] Akgul, M., "A Genuinely Polynomial Primal Simplex Algorithm for the assignment problem", *Discrete Applied Mathematics*, Vol. 45, 1993, pp. 93-115.
- [4] Arkin, Ronald C., "Cooperation Without Communication: Multiagent Schema-Based Robot Navigation," *Journal of Robotic Systems* 9(3), 1992, pp. 351-364.
- [5] Brooks, Rodney A. and Flynn, Anita M., "Fast, Cheap and Out of Control: A Robot Invasion of the Solar System," *Journal of the British Interplanetary Society*, Vol. 42, 1989, pp. 478-485.
- [6] Brooks, Rodney A., "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, March 1986, pp. 14-23.
- [7] Chen, Qin and Luh, J.Y.S., "Coordination and Control of a Group of Small Mobile Robots," IEEE International Conference on Robotics and Automation, Vol. 3, 1994, pp. 2315-2320.
- [8] Chvatal, Vasek, *Linear Programming*, New York: W.H. Freeman and Company, 1980.
- [9] Edmonds, "Maximum Matching and a Polyhedron With 0,1-vertices," J. Res. Nat. Bur. Standards," Vol. 65B, 1965, pp. 125-130.
- [10] Gabow, "Data Structures for Weighted Matching and Nearest Common Ancestors With Linking," Proceedings of the First ACM/SIAM Symposium on Discrete Algorithms, 1990, pp. 434-443.
- [11] Gabow, H., "An Efficient Implementation for Edmonds' Algorithm for Maximum Matching on Graphs," *Journal of the Association for Computing Machinery*, Vol. 23, No. 2, April, 1976, pp. 221-334.
- [12] Gabow, H., "Implementations of Algorithms for Maximum Matching on Nonbipartite Graphs," PhD Thesis, Computer Science Department, Stanford U., 1973.
- [13] Ignizio, J. and Cavalier, T., *Linear Programming*, Prentice Hall, Englewood Cliffs, N.J., 1994.
- [14] Kube, Ronald C and Zhang, Hong, "Collective Robotics: From Social Insects to Robots," *Adaptive Behavior*, Vol. 2, No.2, pp. 189-218.

- [15] Leeuwen, J., "Graph Algorithms," in *Handbook of Theoretical Computer Science, Volume A*, J. van Leeuwen, editor, MIT Press, Cambridge Massachusetts, 1990, pp. 525-632.
- [16] Micali, S. and Vazirani, V., "An $O(\sqrt{VE})$ Algorithm for Finding Maximum Matching in General Graphs," Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science, 1980, pp. 17-27.
- [17] Noreils, Fabrice R., "Toward a Robot Architecture Integrating Cooperation Between Mobile Robots: Application to Indoor Environment," *The International Journal of Robotics Research*, Vol. 12, No. 1, February 1993, pp. 79-98.
- [18] Orlin, J., "A Polynomial-Time Primal Network Simplex Algorithm for Minimum Cost Flows," Proceedings of the 7th Annual ACM/SIAM Symposium on Discrete Algorithms, 1996, pp. 474-481.
- [19] Reynolds, Craig W., "Flocks, Herds, and Schools," *Computer Graphics*, Volume 21, No. 4, July 1987, , pp. 25-34.
- [20] Sokkalingam, P., Sharma, P., and Ahuja, R., "A New Primal Simplex Algorithm for Network Flow Problem," Presented at NET-FLOW'93, to appear in Mathematical Programming B.
- [21] Yamaguchi, Hiroaki, and Arai, Tamio, "Distributed and Autonomous Control Method for Generating Shape of Multiple Mobile Robot Group," IEEE International Conference on Intelligent Robots and Systems, Vol. 2, 1994, pp.800-807.
- [22] Yoshida, E.; Arai, Tamio, Ota, Jun; Miki, Tomoyoshi, "Effect of Grouping in Local Communication System of Multiple Mobile Robots," IEEE International Conference on Intelligent Robots and Systems, Vol. 2, 1994, pp. 808-815.