Optimal Coverage of a Tree with Multiple Robots

Israel Aldana-Galvan, Juan Carlos Catana-Salazar,
José-Miguel Díaz-Báñez, Frank Duque, Ruy Fabila-Monroy,
Marco A. Heredia, Adriana Ramírez-Vigueras and Jorge Urrutia

March 25, 2020

Optimal coverage of a tree with multiple robots

I. Aldana-Galván*[4], J.C. Catana-Salazar†[4], J.M. Díaz-Báñez‡[2], F. Duque§[6], R. Fabila-Monroy¶[3], M.A. Heredia‖[5], A. Ramírez-Vigueras**[1], and J. Urrutia††[1]

[1]Instituto de Matemáticas, UNAM, Mexico City, Mexico
[2]Departamento de Matemática Aplicada II, Universidad de Sevilla, Spain
[3]Departamento de Matemáticas, CINVESTAV, Mexico City, Mexico
[4]Posgrado en Ciencia e Ingeniería de la Computación, UNAM, Mexico City, Mexico
[5]Departamento de Sistemas, UAM-Azcapotzalco, Mexico City, Mexico
[6]Instituto de Matemáticas, Universidad de Antioquia, Medellín, Colombia

Abstract

We study the algorithmic problem of optimally covering a tree T with k mobile robots. The tree is known to all robots, a robot moves along the edges of T. The time and cost of traveling an edge of T is unitary. Our goal is to design a covering strategy in which every vertex of T is visited by at least one robot. This is achieved by asigning to each robot a walk. Two objective functions are considered: the cover time and the cover length. The cover time is the maximum time a robot needs to finish its assigned walk; the cover length is the sum of the lengths of all the walks. We also consider a variant in which the robots must rendezvous periodically at the same vertex in at most a certain number of moves. We show that the problems are essentially different for both cost functions. Some variants of our problems can be solved in polynomial time while others are NP-hard. A summary of our results is shown in Figure 1.

1 Introduction

Terrain coverage is a crucial task to many robotic applications, such as search-and-rescue, lawn mowing, surveillance by unmanned aerial vehicles ([2], [1]), just to name a few. Evidently, coverage can be sped up with the use of multiple robots, in which coverage path planning is performed by a set of robots. In these scenarios, the environment can be modeled by a geometric structure, represented as the union of polygonal obstacles, or a graph structure. The former model assumes that the robots know everything within their sight. For the latter model, the terrain is partitioned into cells inducing a graph whose nodes correspond to locations in the cells, and edges correspond to paths between the locations. In this paper we consider the graph model in which the underlying graph is a tree obtained from a triangulation of a simple polygon whose interior has to be guarded. Our guards must then traverse the edges of the dual graph of the triangulation which is a tree. In fact, spanning trees have been frequently used in multi-robot coverage problems [5].

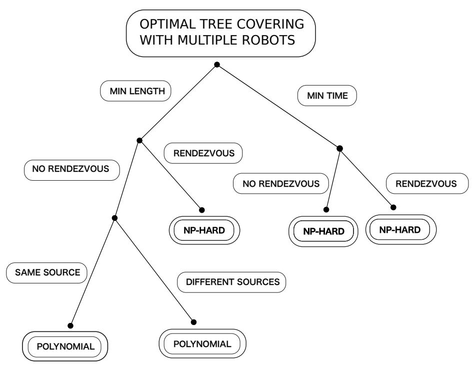

Figure 1: Complexity of the tree covering optimization problems.

Choset [3] provides a survey of coverage algorithms, which distinguishes between off-line algorithms, in which a map of the work-area is given to the robots, and on-line algorithms, in which no map is given. Two variants can also be considered according to the movement cost: in the first one, called uniform, the cost of moving a robot from a region to a neighboring one takes unit time and, non-uniform, otherwise. In this paper we work on the off-line/uniform-cost scenario.

*Email: ialdana@ciencias.unam.mx
†Email: j.catanas@uxmcc2.iimas.unam.mx
‡Email: dbanez@us.es. This research has received funding from the projects GALGO (Spanish Ministry of Economy and Competitiveness, MTM2016-76272-R AEI/FEDER,UE) and CONNECT (EU-H2020/MSCA under grant agreement 2016-734922).
§Email: rodrigo.duque@udea.edu.co
¶Email: ruyfabila@math.cinvestav.edu.mx
‖Email: hvma@correo.azc.uam.mx
**Email: adriana.rv@im.unam.mx
††Email: urrutia@matem.unam.mx. Partially supported by Proyecto PAPIIT IN102117 from the Universidad Nacional Autónoma de México.

We also consider the variant in which the robots are required to meet at most every p units of time, for some fixed positive integer p. We assume all robot rendezvous at the same vertex. This rendezvous version is motivated by papers such as [6].

Regarding objective functions, it is frequently desirable to minimize the time in which coverage is completed. In this case, the multi-robot coverage problem refers to computing a walk for each robot so that the *cover time* is minimized. However, the energy efficiency of the walk of a robot can also take into account the distance it travels. Thus in this paper we consider the *cover length*, that is the sum of the lengths of the walks of all robots needed to cover the tree. Note that these cost functions are different because in some cases, when a robot may stay put at a vertex, the overall cover time can increase but not the cover length.

A *covering strategy* is an assignment of a walk to each robot such that every vertex is visited at least once. We consider the following optimization problems according to the tree/off-line/uniform-cost/rendezvous framework:

- **Minimum Length Covering Problem (MLCP)** Find a covering strategy of minimum length with k robots, each starting at a possible different position.
- **Minimum Length Covering Problem with Rendezvous (MLCPR)** Given an integer p, find a covering strategy of k robots with minimum length such that the robots are required to rendezvous in at most p units of time. The robots start each at a possible different position.
- **Minimum Time Covering Problem (MTCP)** Find a covering strategy of minimum time for k robots all starting at the same position.
- **Minimum Time Covering Problem with Rendezvous (MTCPR)** Find a covering strategy of minimum time for k robots all starting at the same starting position, in which the robots rendezvous at most every p units of time, for a given positive integer p.

In the graph model, most of the optimization problems stated above are NP-hard [7]. In this paper we show that the complexity of the above optimization problems is essentially different for the two objective functions, length and time. Our results are summarized in Figure 1.

2 Notation

Suppose we are given a tree T that has to be covered by k identical robots modeled by moving points.

We assume that the robots may occupy the same vertex of T at a given time without colliding, and that they do not block each other. We also assume the robots can move along the edges of T, and that traversing an edge takes an unit of time. At any given unit of time, a robot can traverse an edge of T, or decide to stay put at a vertex. Finally, we assume that all robots share an internal synchronized clock.

We represent the journey made by a robot by a sequence $W := (u_1, \ldots, u_m)$ of vertices of T where two consecutive elements are either adjacent or equal. When they are equal the robot did not move remaining at the same vertex. For convenience, we assume that the robot stop at the last vertex of the sequence. We refer to such a sequence as a *walk* in T.

Similarly, a *path* in this paper is a walk in which each vertex is visited once. However, a robot may stay at a given vertex for a certain time before moving to the next vertex. We denote the set of vertices of a tree T (respectively a walk W) as $V(T)$ (respectively $V(W)$).

The *time* of W, $t(W)$, is the number of steps the robot needs to carry out the journey described by W. Formally, it is the number of terms in W minus one. The *length* of W, $l(W)$, is the number of times the robot changes position.

A *strategy* is a tuple $S := (W_1, \ldots, W_k)$ of k walks on T, where W_i is the walk assigned to the i-th robot, $1 \leq i \leq k$. We say that S is a *covering* strategy if every vertex of T is in some W_i.

The *time* of a strategy S, $t(S)$, is the total time it takes for the robots to carry out the journey described by their assigned walks. Since the robots move in parallel we have:

$$t(S) = \max_{1 \leq i \leq k} \{t(W_i)\}.$$

The *length* of a strategy S, $l(S)$, is the sum of the lengths of its walks. Thus,

$$l(S) = \sum_{i=1}^{k} l(W_i).$$

The robots rendezvous at time i if they are all at the same vertex at time i, that is if the i-th term of all the walks is the same.

3 Minimum Length Covering Problem (MLCP)

The next result enables us to translate the problem of finding a minimum length strategy for T with k robots to the problem of finding a set of k paths of T.

Observe that the set of vertices covered by the i-th robot induce a subtree T_i of T. It is easy to see that each edge of T_i is traversed once or twice. The edges traversed once form a path from the initial position of the i-th robot to its final destination. Thus we have:

Lemma 1 *In an optimal covering strategy* $S := (W_1, \dots, W_k)$ *which minimizes the total length traversed by the robots, the set of edges of each* W_i *can be decomposed into a path* $P(W_i)$ *and a forest* $F(W_i)$*. The edges in* $P(W_i)$ *are traversed once, and those in* $F(W_i)$ *twice.*

Suppose now that $S := (W_1, \dots, W_k)$ is a minimum length covering strategy for T. Let e be an edge of T. By Lemma 1 we have the following. If e is in exactly q of the paths $P(W_i)$, then it is visited exactly q times. If e is not in one of the paths $P(W_i)$, then it is visited exactly twice. Therefore, the cost of S is given by

$$l(S) = \sum_{i=1}^{k} l(P(W_i)) + 2\left|V(T) \setminus \bigcup_{i=1}^{k} V(P(W_i))\right| \quad (1)$$

Observe that the cost of S in Equation 1 only depends on the paths $P(W_i)$. Suppose that $(P_1, \dots, P_k)$, is a tuple of k paths in T. Then we can construct in $O(n)$ time a covering strategy $S = (W_1, \dots, W_k)$ for T that satisfies Lemma 1 and such that for every $1 \le i \le k$, $P(W_i) = P_i$. In particular, this implies that to find the minimum covering strategy for T with k robots, it is sufficient to find a tuple of k paths $(P_1, \dots, P_k)$ that minimizes the following expression.

$$\sum_{i=1}^{k} l(P_i) + 2\left|V(T) \setminus \bigcup_{i=1}^{k} V(P_i)\right| \quad (2)$$

3.1 All Robots Starting at the Same Vertex

First we consider the case when all robots start at the same vertex u. For the purposes of this section we assume that T is a tree rooted at u. By the previous remarks, it is enough to find a tuple $Q = (P_1, \dots, P_k)$ of paths, all starting at u, that minimizes (2). Note that, in particular, the fact that all the P_i start at u, implies that every P_i ends at a leaf of T.

We show a sketch of the algorithm based on dynamic programming to find Q. Assume that the children of every vertex of T are listed in some arbitrary order. For every vertex v of T, let T_v be the subtree of T rooted at v. For every $1 \le i \le \text{degree}(v) - 1$, let $v(i)$ be the i-th child of v; and let $T_v[i]$ be the subgraph of T consisting of the union of the subtrees rooted at the first i children of v and the edges joining these children with v.

We use two tables $C[v, i, j]$ and $P[v, i, j]$, where v runs over all vertices of T, $1 \le i \le \text{degree}(v)$, and $0 \le j \le k$. The table $C[v, i, j]$ stores the minimum cost given by (2) for $T_v[i]$ of a tuple of j paths all starting at v. The table $P[v, i, j]$ stores this tuple, where each path is stored as a linked list of vertices, with a pointer to the first vertex in the path. This representation enables us to concatenate paths in constant time. Note that $P[u, \text{degree}(u), k]$ is our desired Q.

For $j = 0$ and every vertex v in T, we set $C[v, i, 0]$ to be equal to twice the number of vertices in $T_v[i]$. Let v be any vertex of T and let $v_1, \dots, v_m$ be its children. Let P be a tuple of j paths achieving $C[v, i, j]$. Let s be the number of paths of P that end at a vertex of T_{v_i}. If $s = 0$, then $C[v, i, j]$ is equal to

$$C[v, i-1, j] + C[v_i, \text{degree}(v_i), 0].$$

If $s > 0$, then $j - s$ paths end at $T_v[i-1]$. Therefore, we have the following formula:

$$C[v, i, j] = C[v, i-1, s] + C[v_i, \text{degree}(v_i), j-s] + j - s. \quad (3)$$

Therefore, if the previous values have been computed, $C[v, i, j]$ and $P[v, i, j]$ can be computed in $O(k)$ time by checking over all possible values of s, and taking the minimum value.

We compute $C[v, i, j]$ and $P[v, i, j]$ bottom up. For all leaves v of T and every $1 \le j \le k$, we set $C[v, 0, j] = 0$; we also set $P[v, 0, j]$ to be a list of j paths each consisting only of the vertex v. Having computed $C[v, i, j]$ and $P[v, i, j]$ for all vertices of height h, we compute these values for the vertices at height $h + 1$. In total this takes $O(nk^2)$ time and we have the following result.

Theorem 2 *Let k be a positive integer, and let u be a vertex of T. Then for every $1 \le i \le k$, a minimum length covering strategy for T with i robots, all starting at u, can be computed $O(k^2 n)$ total time.*

In a similar way we can prove the next result which will be used in the next section:

Theorem 3 *Let k be a positive integer and let x_1, x_m be vertices of T. We can find in $O(nk^2)$ time a tuple of k paths such that all of them start at x_1, at least j of them end at x_m, and minimize (2).*

3.2 Robots starting at different vertices

We first consider the case in which each robot starts at one of two given vertices. Suppose that in an optimal covering strategy for T, a robot starts at x_1 and eventually visits the node x_{i+1}. Let $x_1 W_1 x_i x_{i+1} W_2$ be the walk made by this robot. Suppose that in such an optimal covering strategy for T another robot starts at x_m and eventually visits the node x_i. Let $x_m W'_1 x_{i+1} x_i W'_2$ be the walk made by this robot. If we replace the walk of the first robot with $x_1 W_1 x_i W'_2$ and the walk of the second robot with $x_m W'_1 x_{i+1} W_2$, we obtain an exploring strategy of smaller length; this is a contradiction. Therefore, every edge (x_i, x_{i+1}) is only traversed by robots in one direction. We exploit this property in our algorithms. Generalizing the dynamic programming approach used when all robots start at the same vertex, we can prove:

Theorem 4 *Let k be a positive integer and let u, v be two vertices of T. We can find in $O(nk^3)$ time a tuple of k paths such that s of them start at u, t paths start at v, $s + t = k$, and they minimize (2).*

Theorem 4 is used to prove that the general problem can also be solved with dynamic programming, as well as the main result in this section:

Theorem 5 *A minimum length covering strategy for T, with k robots at arbitrary starting positions, can be computed in $O(k^3 n + 2^k k^{k+1})$ time.*

4 Minimum Length Covering Problem with Rendezvous (MLCPR)

In this section we study the following problem:

Problem 6 (LCSR) *Find a continuous covering strategy of T such that the robots have to rendezvous repeatedly, the time between consecutive rendezvous is at most p, and the distance traveled by the k robots between consecutive rendezvous, $l(S)$ is at most l.*

We can prove that LCSR problem is NP-Complete by a reduction from 3-PARTITION. This problem is known to be strongly NP-complete [4]. The following result can be proved:

Theorem 7 *The LCSR Problem is NP-complete.*

Proof. (Sketch) Whether a given strategy satisfies that the robots rendezvous every p steps can be verified in polynomial time. Since the length of a given strategy also can be computed in polynomial time, we have that LCSR is in NP. Let (A, B) be an instance of 3-PARTITION such that $|A| = 3m$ and B is bounded by a polynomial on m. We construct in polynomial time an instance of LCSR that has a solution if and only if A admits a 3-PARTITION.

Let $A := \{a_1, \dots, a_{3m}\}$. Let $P_1, \dots, P_{3m}$ be m paths, all starting at the same vertex u, such that $l(P_i) = a_i$. Let

$$T' = \bigcup_{i=1}^{m} P_i.$$

Let $Q = (v_1, \dots, v_{3B+1})$ be a path of length $3B$. Let

$$T := T' \cup Q \cup (u, v_1).$$

Note that since B is bounded by a polynomial on m, T can be constructed in polynomial time. Let $k := m + 1$, $p := 2B$ and $l = 4mB + 2B + 2m$.

We claim that T has a covering strategy, with k robots starting at v_1, that rendezvous at most every p steps, and of length at most l if and only if A admits a 3-partition. □

5 Minimum Time Covering Problem (MTCP)

The problem of computing a Minimum Time Covering Strategy is NP-hard, regardless if periodic rendezvous is required. The corresponding decision problem (Time Covering Strategy, TCS) is as follows:

Problem 8 (TCS) *Let T be a tree, with k robots at given starting positions. Let t be a positive integer. We want to know if there exists a covering strategy S for T with these robots so that $t(S)$ is at most t.*

The TCS problem can also be shown to be NP-Complete by a reduction from 3-PARTITION.

Theorem 9 *The TCS Problem is NP-complete.*

References

[1] J. J. Acevedo, B. Arrue, J. M. Díaz-Báñez, I. Ventura, I. Maza, and A. Ollero. One-to-one coordination algorithm for decentralized area partition in surveillance missions with a team of aerial robots. *Journal of Intelligent & Robotic Systems*, 74(1-2):269, 2014.

[2] E. M. Arkin and J. SB. Fekete, S. P .and Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(1):25–50, 2000.

[3] H. Choset. Coverage for robotics–a survey of recent results. *Annals of mathematics and artificial intelligence*, 31(1-4):113–126, 2001.

[4] M. R. Garey and D. S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM J. Comput.*, 4(4):397–411, 1975.

[5] N. Hazon and G. A. Kaminka. On redundancy, efficiency, and robustness in coverage for multiple robots. *Robotics and Autonomous Systems*, 56(12):1102–1114, 2008.

[6] G. A. Hollinger and S. Singh. Multirobot coordination with periodic connectivity: Theory and experiments. *Robotics, IEEE Transactions on*, 28(4):967–973, 2012.

[7] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser. Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research*, 34:707–755, 2009.