Solving sweeping problem for trees in graph theory

W.A.L. NIWANTHI[1], H.D. PANDITHARATHNE[2], S.S.N. PERERA[3]

waniw@ou.ac.lk

[1] Department of Mathematics & Philosophy of Engineering,
The Open University of Sri Lanka, Nawala, Nugegoda
[2] Department of Mathematics, University of Colombo, Colombo 03
[3] Research & Development Center for Mathematical Modelling, Department of Mathematics,
University of Colombo, Colombo 03

We develop a theory to determine the search number of a graph that allows us to detect an intruder along an edge without limiting the visibility of adjacent vertices. The presented technique here will allow to express the sweep problem as a linear program using an existing formulation of a linear program designed for problems where capture occurs only at a vertex of a graph. We also provide a method to solve the sweep problem for any complex tree, utilizing a set of sub-trees of the tree.

Keywords: Search and sweep problem, set covering problem, branch cut algorithm, homeomorphic trees.

DOI: 10.5604/01.3001.0054.6288

1. Introduction

Searching for a lost man or an object in an environment in a real-world context would be a matter of collective efforts of man power and use of secondary operations such as the use of robotics or technology driven solutions. Let's name the said lost person as an intruder for the context, which could be a continuously moving object or could be a person who keeps on taking intelligent moves to avoid being caught or might be an object which is intangible. In mathematical context and in general, the idea that one group of people will be tracking down or pursuing a single or multi-person party in an environment while another party will be trying to evade the former are called pursuit-evasion problems. The search and sweep problems in graphs were initially proposed by the mathematician T.D. Parsons in his article [1].

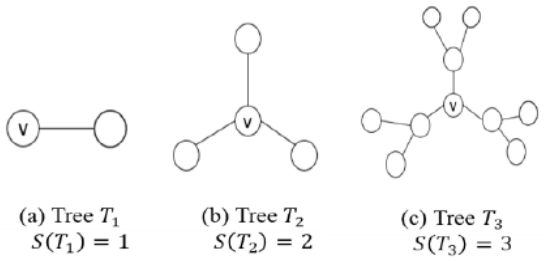

Fig. 1. Optimal trees for search numbers 1, 2, and 3

His findings were developed specifically for tree structures. He introduces a set of optimal tree structures $\{Tk\}$ (see Figure 1) in [1] for each search number and produces an algorithm to find the search number for any arbitrary tree structure, where the minimum amount of people needed for the search party is called the search number.

Throughout this project, we denote the search number of graph G by S(G).

Countless variants of pursuit-evasion problems exist in literature, in discrete and in continuous forms. Here we will consider a discrete pursuit-evasion problem with tree-structured graph environment, which is called the Sweep Problem. The said tree will be finitely connected without loops or multiple edges. A vertex would be a connection point of two or more edges. The notion behind this is, the intruder and searchers will be moving along the edges and vertices of a graph. Their behavior is constrained by the structure of the graph.

The idea is, if we take a cave system blueprint as the environment where the person got lost, in the sweep problem, searchers and the intruder will meet each other at any place inside of the cave. It could be either on a vertex or on an edge in the graph theory context.

The searching party have some restrictions during the search operation. But the intruders have no such restrictions. From the survey done by B. Alpach in [2], he sets several sweeping

models. One searcher could be moving at one time interval while rest of the searchers are on hold, or the searchers will be moving in a discontinuous way while intruder keeps moving according to some continuous function.
Other than conducting search operations for intruders in a cave system, some other applications has been introduced [3] explains application in mobile robotics motion planning, in telecommunication network systems etc. Further clearing a complex system of connected pipes to remove some noxious gas, finding a lost vehicle in a particular road system were discussed in [4], [5], and searching a computer network to find a virus has been given in [6] as real-life applications associated to our subject area. Using concepts of robotics and graph theory for indoor pursuit evasion to catch a mobile intruder and providing an algorithm to catch an intruder who actively avoid searchers, node searching when the intruder's speed is infinite which discussed in [7] are some interesting illustrations related to this entire topic of discussion. Although finding the search number for general graphs are NP-hard problems as in [8], several search algorithms with different complexity results are explained in [9] by Borie. According to A.S. LaPaugh [10] and Sheng--Lung Peng [11], the search number of a tree can be computed in linear time with a time complexity of $O(n2 \log n)$. In relation to the combinatorial approaches conducted so far on the subject, one popular approach is a mixed integer programming approach which heavily dedicates itself for search operations on general graphs. Our study itself has focused on initial presentations brought by Z.C. Taskin et al., who elaborates on node searching in [12]. They provide an exponential-size set-covering formulations for these problems and use the dual problem of the formulation to introduce a linear program along with the branch-cut-price algorithms to solve them.

On a similar note, multi agent path-finding approach on trees also brought out into literature which also uses branch-cut-price algorithm in solving the combinatorial problem in [13]. Models brought out so far haven't directly addressed the linear/integer programming formulation to find the search number needed to clear a tree under sweep problem. Hence an attempt is brought out in this study, which further have room for improvement. The paper is organized as follows. Section 2 discusses how to add some vertices to the graph in order to cooperate the set covering formulation and hence the linear programming formulation that defined in [12] by Z.C. Taskin et al., for the search problem (they assume the visibility of adjacent vertices of the graph) in to the sweep problem. In Section 3, it is explained how to determine a sub-tree structure in order to obtain the search number and then how to use sub-tree structure to solve the sweep problem for complex tree structures. Following that section 4 will describe how to apply findings to a sweep problem for an arbitrary tree structure. This section also includes a concrete example.

2. Linear program for sweep problem

The linear programming formulation to find the search number in a general graph introduced by J. Cole Smith et al. [12] using the following set covering formulation. They denote the set of all possible walks of length T by $P(T)$, where a walk $p \in P(T)$ is a sequence of vertices $\{i_1, i_2,\ldots, i_r\}$ of the graph such that for all $1 \le k \le r-1$, i_{k+1} is an adjacent node of node i_k of the graph. Also they define the length of a walk as the number of edges in the walk.

Model (page 5, [12])

$$\text{minimize} \quad \sum_{p\in P(T)} \lambda_p \tag{1}$$

$$\text{subject to} \quad \sum_{p\in P(T)} d_{pr}\lambda_p \ge 1 \tag{2}$$

$$\lambda_p \in (0,1),\ d_{pr} \in (0,1)$$

where:

λ_p = Binary variable which equals 1 if a searcher is assigned to follow the walk p
d_{pr} = Binary parameter which equals 1 if the intruder following the walk r gets caught to the searcher which follow the searcher walk p.

The objective function (1) minimize the number of searchers traversing in the graph, and the sets of constraints given by (2) ensure that for each possible intruder route, there is at least one searcher who can detect it. We identified that the same set cover formulation can be used for sweep problem which allows to meet the intruder to a searcher even along an edge, not only at a vertex as assumed in [12] with the following modification of the graph. And hence the linear program given in page 8 in [12].

Here we modify the graph by adding an additional vertex on each edge of the graph by considering them as a place where searcher and intruder can meet. Addition of those vertices

will eliminate the visibility assumptions of the adjacent vertices of the graph given in [12].

3. Use of the search number of subtrees to find the search number for complex trees

Article [12] used the branch-cut-price algorithm to the whole graph to find the search number in a given arbitrary graph within a given time period. The complexity of the problem is higher due to their consideration of the full graph. So, here we propose to use the results of Parson in [1] to find the search number for trees. Here, we deconstruct the provided tree structure into its constituent homeomorphic disjoint copies. Homeomorphism of tree structures indicate the same tree that is drawn differently. But those are considered as equivalent since the search number does not change.

Then we use the following two theorems given by Parson in [1] to simplify our operations.

Theorem 1 (page 4 in [1]).
Let k be an integer such that $k \geq 1$ *and let T be a tree. Then* $S(T) \geq k + 1$ *if T has vertex v at which, there are at least three branches* T_1, T_2, T_3 *satisfying* $S(Tj) \geq k$ *for* $j = 1, 2, 3$.

Theorem 2 (page 5 in [1]).
Let $k \geq 2$. *Let T be a tree. Then* $S(T) = k$ *if and only if T has a subtree homeomorphic to a tree in* T_k, *but T has no subtree homeomorphic to a tree in* T_{k+1}.
Theorem 2 refer $\{T_k\}$'s as set of recursive set of trees for each search number which introduced by Parson such that all the trees in set $\{T_k\}$ are homeomorphic. Let's refer them as optimal tree structures introduced by Parson. Figure 1 gives some example graphs from optimal homeomorphic tree structures of T_1, T_2, T_3 and T_4. In this paper we will be referring T_k's as a set of homeomorphic maximal tree structures of a given tree. Also we will be using the set cover formulation and optimal trees T_ks, Parson introduced which the search number is known, to find the search number of these homeomorphic subtree clusters T_k.

3.1. Reduction of Homeomorphic Clusters

Discovering the maximal tree structures is a way of characterizing the structure of a tree. Let T be a tree and v be a vertex of T. A branch starting from vertex v is called a maximal subtree of T according to [1], if it supports the condition that the valence of v is one in subtrees (see Figure 2).

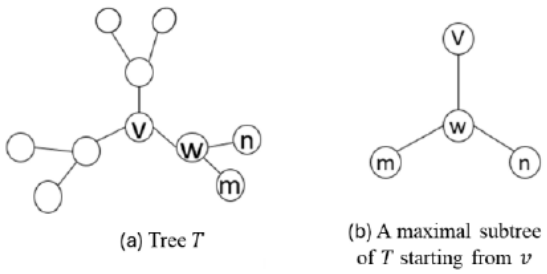

Fig. 2. Maximal subtree

A cluster of a selected tree can be identified as a disjoint graph taken from a root of a tree. On a recurrent sub trees T_k introduced by Parson, clusters made from the root node will have same adjacency matrices which will highlight their equivalence relation. And, all maximal tree clusters will be equal in size for those T_k subtrees. But in our paper, the clusters $\{T_k\}$ will not to be in the same size. The cluster reduction will happen by taking a single vertex where there exist maximum number of clusters which are maximal subtrees.

4. Sweep a Tree

Let T be a tree which needs to find the intruder within a finite time. Then the following steps will explain how to sweep a tree by combining all the findings of previous sections:

1. Reduce the tree T to homeomorphic clusters $\{Tk\}$ which are also maximal subtrees.
2. Select a cluster and introduce a vertex for each edges of the cluster as suggested in section 2 to address the possibility of capturing the intruder on an edge. 2 to address the possibility of capturing the intruder on an edge.
3. Decide the time period which we can spend on finding the intruder if it is not given. Time period needed for the search operation are taken into consideration according to the number of edges the searcher or the intruder travel prior to getting caught. In the sweep problem, we assume both parties are traversing from one vertex to the adjacent vertex without omitting any vertices. Hence the number of time steps equals the number of edges in a particular search path. To be more specific, one-time step of the search operation is defined as, one step going from one vertex to the other. So the time period that we are planning to spend can be

decided according to the number of time steps that we proceed. If the number of time steps is not specified, we suggest considering the number of edges that can connect the root node to the farthest leaf vertex. Since that gives the minimum number of time steps needed if the searcher moves forward without backtracking.

4. Find the search number for the particular cluster. We will use pre-identified optimal trees T_k defined by Parson in [1] to find the search number of the cluster. Note that T_k is defined for all the search numbers by Parson in [1]. If the cluster is not in any of optimal tree structures T_k, we will use the branch-cut-algorithm to find the search number of the cluster. The search operation happens by clearing one cluster at a time. Hence there's a possibility for the moving intruder to move past clusters where searchers are occupied. In order to make sure that the moving intruder doesn't jump across the clusters while mobile searchers are searching a certain cluster, we have to place stationary searchers on the nodes where cluster reduction happens. Therefore, a stationary searcher will be placed at node v. Hence, total search number of the reduced cluster equals the mobile searchers in the homeomorphic cluster plus the stationary searcher positioned at the starting point of the reduced cluster.
5. Follow the same steps from 2 to 4 for other remaining clusters. Then the search number of the original tree T will be $S(T_k)$ when the cluster T_k gives the maximum search number which is equal to $S(T_k)$ among all the clusters.

4.1. Demonstration of the findings

Let T be given as in Figure 3(a). Then let us consider the reduced cluster which is given in Figure 3(b). Once the cluster is reduced, we add temporary vertices x, y, z on each edge as in Figure 3(c) and take those vertices as the meeting points of the searcher and the intruder on the edge.

Here we can use the reduced homeomorphic cluster (b) of T which given in Figure 3 is in the optimal tree structures T_2 which introduced by Parson. Therefor according to [1], the graph (b) has search number 2 (see Figure 1) and hence we need 2 mobile searchers to cover the reduced cluster (b). Therefore, the search number of reduced cluster will be 3 with the stationary searcher at the vertex v.

Note that we also can use branch-cut-algorithm to the graph Figure 3(c) with time step equal to 4 since there are 4 edges from vertex v to the any of leaf vertex m or n.

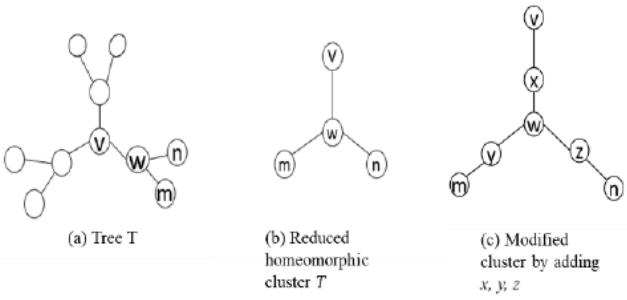

Fig. 3. Homeomorphic cluster reduction and addition of vertices

Additional to the cluster that we considered in Figure 3(b), we can see there are two more clusters at vertex v on tree T. One is homeomorphic to the cluster Figure 3(b) that we already considered, so it has the same search number 3. And the other cluster connects only one vertex to v on tree T. So, it belongs to T_k and hence has the search number equal to 2.

Then since the maximum search number is 3 which is required to cover the clusters, the search number of tree T is equal to 3.

5. Conclusion

In this study, we explain how to determine a graph's search number where the intruder can be located along edges as well as vertices. Additionally, this takes care of the situation where neighboring vertices are obscured. We were able to apply the branch-cut approach to get the search number in any given graph that included edges as meeting points since the changes made by adding more vertices to each edge of the graph still permitted to describe the problem as a linear program. Although adding more vertices increases the number of steps, it decreases the amount of time required to clean the network.

As a special situation, we concentrate on trees and use a pre-defined collection of optimal trees to determine the search number of subtrees of the original tree. In the absence of an ideal tree that is homomorphic to the subtree, the branch-cut algorithm is applied.

As future works, image processing can be used to enhance the computational programming portion. The study can also be applied to scenarios in which searchers or intruders start their operations at random points on the graph

rather than at the root vertex or leaf. The research could further explore the notion of discretizing edges and could assist in circumstances when the pace of the intruder and the searcher is varied.

6. Bibliography

[1] Parsons T.D., "Pursuit-evasion in a graph", [in:] *Theory and applications of graphs*, p. 426–441, Springer, 1978.

[2] Alspach B., Dyer D., Hanson D., Yang B., "Time constrained graph searching", *Theoretical Computer Science*, 399(3), 158–168 (2008).

[3] Fomin F.V., Golovach P.A., Kratochvıl J., "On tractability of cops and robbers game", Fifth IFIP International Conference on Theoretical Computer Science – TCS 2008", IFIPAICT, vol. 273, 171–185, Springer, 2008.

[4] Alspach B., "Searching and sweeping graphs: A brief survey", *Le matematiche*, 59(1, 2), 5–37 (2004).

[5] Penuel J., Smith J.C., Shen S., "Models and complexity analysis for the graph decontamination problem with mobile agents", *Networks*, vol. 61(1),1–19 (2013).

[6] Chung T.H., Hollinger G.A., Isler V., "Search and pursuit-evasion in mobile robotics", *Autonomous robots*, vol. 31(4), 299–316 (2011).

[7] Kehagias A., Hollinger G., Singh S., "A graph search algorithm for indoor pursuit/evasion", *Mathematical and Computer Modelling*, vol. 50(9-10), 1305–1317 (2009).

[8] Megiddo N., Hakimi S.L., Garey M.R., D.S Johnson, Ch.H. Papadimitriou, "The complexity of searching a graph", *Journal of the ACM* (*JACM*), vol. 35(1), 18–44 (1988).

[9] Borie R., Tovey C., Koenig S., "Algorithms and complexity results for graph-based pursuit evasion", *Autonomous Robots*, vol. 31(4), 317–332 (2011).

[10] LaPaugh A.S., "Recontamination does not help to search a graph", *Journal of the ACM* (*JACM*), vol. 40(2), 224–245 (1993).

[11] Sheng-Lung Peng, Chin-Wen Ho, Tsan-Sheng Hsu, Ming-Tat Ko, Chuan Yi Tang, "Edge and node searching problems on trees", *Theoretical Computer Science*, vol. 240(2), 429–446 (2008).

[12] Caner Taşkin Z., Cole Smith J., "Branch--cut-price algorithms for solving a class of search problems on general graphs", *Networks*, vol. 70(1), 4–18 (2017).

[13] Lam E., Le Bodic P., Harabor D., Stuckey P.J., "Branch-and-cut-and-price for multi-agent path finding", *Computers & Operations Research*, vol. 144, 105809 (2022).

Rozwiązywanie problemu przeczesywania drzew w teorii grafów

W.A.L. NIWANTHI, H.D. PANDITHARATHNE, S.S.N. PERERA

Opracowano teorię wyznaczania liczby przeszukiwań grafu, która pozwala wykryć intruza wzdłuż krawędzi bez ograniczania widoczności sąsiednich wierzchołków. Przedstawiona technika pozwoli wyrazić problem przeczesywania grafu w postaci zadania programowania liniowego, wykorzystując istniejące sformułowanie programu liniowego przeznaczonego dla problemów, w których przechwytywanie następuje tylko w wierzchołku grafu. Przedstawiono również metodę rozwiązywania problemu przeczesywania dla dowolnego złożonego drzewa, wykorzystując zestaw poddrzew drzewa.

Słowa kluczowe: problem wyszukiwania i przeczesywania, problem pokrycia zestawu, algorytm wycinania gałęzi, drzewa homeomorficzne.