Solving Pursuit-evasion Problems with Graph-Clear: an Overview

Andreas Kolling and Stefano Carpin

***Abstract*— This paper presents an overview of the numerous results we recently published for the problem of multi-robot pursuit evasion. We review the Graph-Clear formalism we introduced, we summarize the variants we studied, and the main results we derived. Finally, we outline directions for future research both in Graph-Clear and for pursuit-evasion and search problems in general.**

I. Introduction

The use of multiple robots for distributed information collection over large areas has seen an explosive growth in the last years. This is the result of astounding progress that has been made in engineering, enabling the development of robust robots to complete such missions, and sound theoretical results in numerous disciplines that enable the design of algorithms to control large teams of robots. Such progress has enabled the consideration of increasingly applicable scenarios, including border patrol, surveillance of sensitive areas, and search and rescue. Despite the progress, however, the problem is far from being solved, and open questions outnumber by far the problems for which a definitive answer is known. Particularly, one pressing demand remains: to develop teams of cooperating agents that are more numerous, individually less powerful, more resilient as a team, and possibly accommodating a sliding degree of autonomy.

Motivated by this we have proposed a formal framework aiming to answer some of the fundamental theoretical questions underpinning above mentioned applications. Our formalism is concerned with a pursuit-evasion scenario that models the detection of an arbitrary intruder in an environment and can provide formal guarantees of capture. Such a model is particularly suitable for security and rescue scenarios in which target movement is a concern. To accomodate the requirements of a multi-robot system *yet another pursuit evasion approach* was in our opinion needed in order to fill the following gaps:

- consider robotic agents that have only limited sensing power. Such robots may be unable to accomplish powerful operations on their own, but may be capable of more sophisticated actions when properly coordinated;
- formalize the problem using a model that can be applied to largely different situations, for example prescinding from the specific details of the operative environment.

These requirements drove the development of Graph-Clear, a graph-based pursuit-evasion problem.

Andreas Kolling is with the School of Information Sciences, University of Pittsburgh, PA, USA.

Stefano Carpin is with the School of Engineering, University of California, Merced, CA, USA.

Any graph-based pursuit-evasion strategy, including Graph-Clear, hinges on the ability to answer two important questions:

- How to obtain suitable graph representations of an environment of interest?
- How to relate abstract graph actions to robot algorithms that are executed by the team of robots?

In our opinion, Graph-Clear finds one of its major strengths in the intuitive (yet powerful) answers it assumes for the above questions.

II. State of the Art for Graph-Clear

In [13] we give precise definitions for Graph-Clear. In summary, Graph-Clear is a problem defined on a surveillance graph. A surveillance graph is an undirected graph $G = (V, E, w)$ where V is a set of vertices, E is a set of edges, and w is a weight function associating an integer cost to edges and vertices. Initially G is entirely contaminated. As commonly assumed in related literature, *contamination* represents the possible presence of an intruder. On the contrary, when it is known that no intruder is located in a certain element (edge or vertex), the component is labeled as *clear*. Contamination is removed using only two action, i.e *sweep* and *block*. A sweep on a vertex clears it. For reasons outline later, we hypothesize that while a vertex is being cleared all its incident edges must be blocked. A block on an edge clears the edge and prevents its recontamination. A strategy is a sequence of multiple actions that can be carried out at every step. Contamination spreads immediately through all vertices and edges that are not blocked and originates from contaminated vertices and edges. Every action has an associated cost given by w and the goal of Graph-Clear is to determine a strategy with minimal cost that clears the graph.

We distinguish two basic types of strategies, those that are required to be contiguous and those that are not, denoted as non-contiguous. Contiguity requires all cleared vertices to form a connected subgraph. In some applications this provides the team with a safe and connected area in which one can move and deploy infrastructure. Contiguous strategies turn out to be simpler to compute [8], [13]. Yet, they are generally more expensive. For edge-searching imposing contiguity can increase the cost of the strategies almost by two-fold [3] and from [8] we can construct graphs for which there is also an increased cost for contiguity for Graph-Clear.

An alternative graph-based pursuit-evasion model that can be applied in a robotic context is edge-searching and the edge-searching variant with a node-located intruders as described in [4]. Applying weights to edge-searching as done in [2] then leads to graph model that also deals with large

numbers of robots and multiple robots per action. The key differences between weighted edge-searching and Graph-Clear are in the requirements imposed for the implementation of basic operations. To apply edge-searching one needs to provide implementations for guarding vertices and sliding along edges, while in Graph-Clear one needs to implement sweeping vertices and blocking edges. An implementation of guarding has to guarantee that all intruders in the associated region for the vertex are detected and furthermore that no intruder can enter or exit the region undetected. Sweeping does not require the latter. Instead, in Graph-Clear while sweeping a vertex we require a block on each edge to prevent targets from entering or exiting. The consequence is that some robot algorithms cannot be used for guarding operations. The example in Fig. 1 uses an algorithm for detecting targets inside the region of a vertex that does not satisfy the guarding requirements and is hence not directly suitable for edge-searching. To satisfy the guarding requirements one would have to augment the algorithm by additionally positioning robots at the entrances. Then the cost of this combined routine becomes a weight in weighted edge-searching which represents the number of the robots needed to search the region and to keep entrances covered. But, once the robots searched the region and hence cleared the vertex we still have to guard the vertex to prevent recontamination of its neighbors. In practice this continued guarding after the actual search does not need to involve any of the robots that performed the search, but only those covering the entrances. But in weighted-edge searching we still pay the full cost for the guarding operation. This is because in edge-searching guarding of a vertex performs two basic functions, namely the prevention of spreading of contamination from and to its neighbors, and additionally the detection of all intruders in the vertex. One could try to overcome this problem by having weights on edges represent the cost of entering a vertex, searching and covering the entrances while the weight on the vertex only represent the cost of covering the entrances. But then sliding along an edge costs more than guarding the vertex. Not only is this unintuitive, but the formulation of weighted edge-searching from [2] does not allow edge weights larger than the weight of the adjacent vertex. Finally, the algorithm presented in [2] turns out not be optimal for the weighted case as shown in [9].

A. Graph-Clear algorithms

One of the major results we identified while studying Graph-Clear concerns its intrinsic computational hardness [13]. In its generality, for a given graph the problem of determining the solving strategy using the least number of robots is NP-hard. This limiting result motivates the study of algorithms on trees that are heuristically obtained from graphs. Given a surveillance graph, the most trivial transformation into a surveillance tree is obtained computing a minimum spanning tree according to a suitably defined cost, and then permanently blocking the edges not part of the spanning tree. It is immediate to realize that these cycle-removing blocks can be lifted as soon as the adjacent vertices

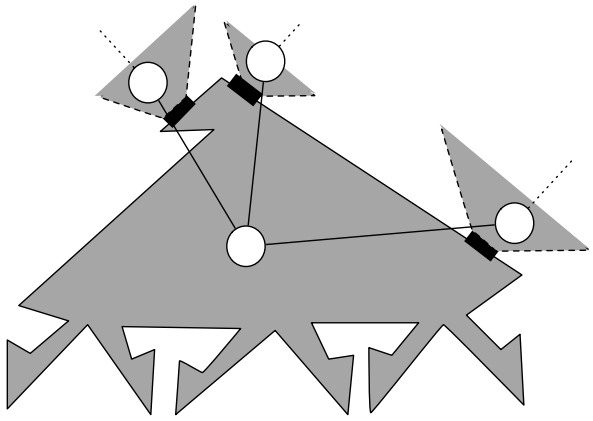


Fig. 1. An example that illustrates how a graph for Graph-Clear can relate to an actual environment. The environment is shown in grey with its graph embedded. All weights in this example are equal to one. Connections between regions that are connected by edges are shown in black. The center region is the "eagle" example redrawn from [16]. It can be cleared using the algorithm from [16] with only one robot and a simple gap sensor with sufficiently large range. During its execution it recontaminates the top part of the region and hence cannot guarantee that no target enters the vertex undetected. We hence need blocks on the edges, i.e. to position sensors on the black regions. Note that the entire environment can be very large so that the sensor only satisfies the large range assumption within a vertex.

are cleared, and one also quickly realizes that permanent blocks can easily lead to suboptimal strategies because of the constant cost they imply. In general, however, the problem of converting a graph into a tree in the most favorable way for a successive Graph-Clear application (possibly coupled with a dynamical reallocation of the blocks) has not been sufficiently explored.

Most of the algorithms we derived utilize a directional label or other structures such as sequences of cuts that are computed on every edge that captures the cost of clearing subtrees rooted beyond the edge. The precise meaning of these quantities is too long to be appropriately described in this short communication, but their significance can be easily related to the pursuit-evasion problem at hand, and, most importantly, can be easily computed. The reader is referred to the provided references for thorough discussions. Currently, the following algorithms are available for Graph-Clear:

- suboptimal label-based contiguous strategies on trees [6], [13] in polynomial time;
- optimal cut-based contiguous strategies on trees [13] in polynomial time;
- suboptimal label-based non-contiguous strategies on trees [8] in pseudo-polynomial;
- suboptimal strategies on graphs from strategies on trees [6] in polynomial time;
- extensions of previous algorithms to strategies for probabilistic actions [10] in polynomial time;
- modification of Graph-Clear that requires a sweep to also prevent recontamination ([5] Chapter 3).

Hence, the problem of computing optimal contiguous strategies on trees in polynomial time is solved. It is currently unclear whether one can compute non-contiguous strategies in polynomial time and we conjecture that this problem is

in fact pseudo-polynomial [8].

In [4] an improved conversion of strategies from trees to graphs is proposed. The method tries many different spanning trees and converts their strategies to the graph, finally selecting the one with lowest cost. This method can be probabilistically complete if the strategies on the trees are computed with certain label-based approaches. The core ideas can be applied to Graph-Clear as well and the method from [6] which only selects the minimum spanning tree can be extended to test many more spanning trees. We conjecture that this also leads to a probabilistically complete algorithm to compute contiguous Graph-Clear strategies on graphs.

The modification of Graph-Clear that requires vertex sweeps to prevent recontamination from Chapter 3 [5] has the advantage that strategies can require less robots by reducing the number of blocks that are needed during a sweep. Note that this is a local reduction and as such a constant difference, i.e. on very large graphs the savings can be marginal. Yet, for smaller graphs or when every robot counts it can be of an advantage if the sweeping routine can guarantee that no intruder passes through the vertex to previously cleared parts. It turns out that much of the formalism and algorithms for trees still hold since in a tree strategy a vertex is always entered from one edge while all other edges go downward towards the leaves. Hence during the sweep the robots blocking the originating edge are not required and those at the edges to subtrees further down in the tree are only required directly after the sweep.

B. Extracting graphs

The question of how to obtain a graph representation is essential to any graph-model of pursuit-evasion if it should be applied to a robotic context. For Graph-Clear we presented heuristics in [7] and [5] that rely on a Voronoi Diagram to identify narrow parts of a two-dimensional environment. In general, the problem of identifying good graphs for an environment is difficult and depends on the actual implementation of the sweep and block actions on a real robot team. As such the graph extractions, often a partitioning of an environment, has to take into account the capabilities of the robots. Fig. 2 shows an example of a partitioning based on the Voronoi Diagram from [7] and the final graph extracted from it.

Examples how to extract graphs while relying less on heuristics are presented in [11], [12], and [5]. Therein an abstract sensor model, a sweep-line, is defined and robots are assumed to cover this line with sensors as it sweeps through and clears an environment. The surveillance graph is constructed by considering multiple such sweep lines and associating them to edges and vertices. The strategy is then used to create a schedule for the movement of all sweep lines. In [5] (Chapter 4) this problem is discussed in detail. For this sensor we conjecture that there are optimal graphs that can be extracted from a two dimensional environment given by a finite set of convex obstacles. Fig. 3 shows an example of a surveillance graph extracted as the dual of the Voronoi Diagram. Extensions of this idea lead to improved graph extraction as discussed in Chapter 4 of [5].

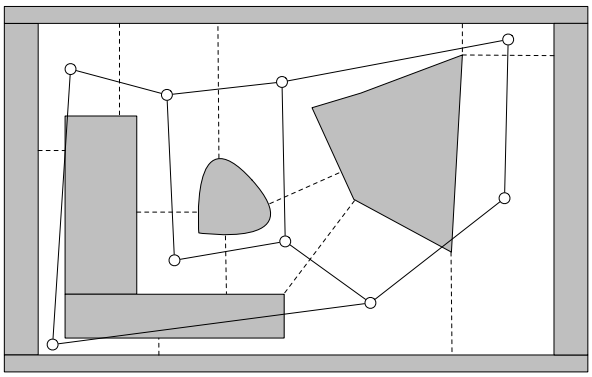
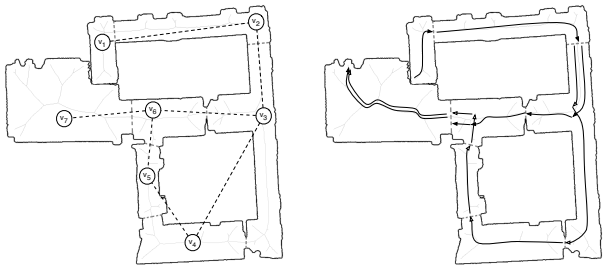


Fig. 3. An example of a surveillance graph created for Line-Clear. Every vertex of the surveillance graph corresponds to a vertex of the Voronoi Diagram.

C. Executing strategies with robot teams

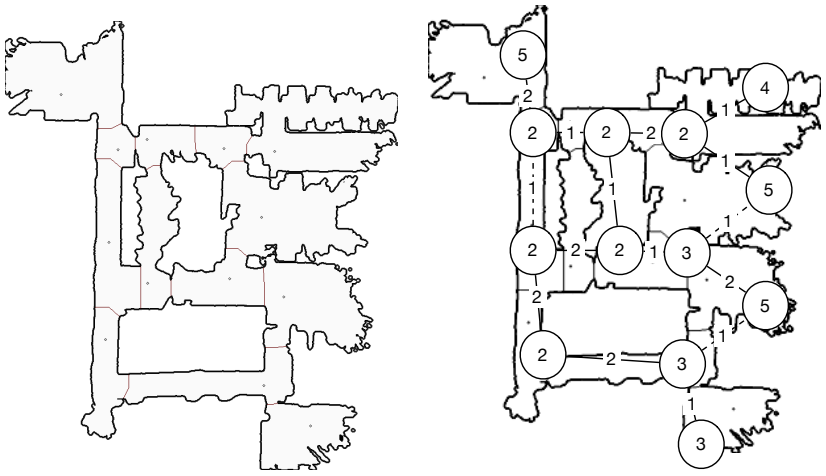
Finally, to demonstrate applicability of Graph-Clear we executed strategies with real and simulated robots. In [5] Chapter 6 experiments with two Pioneer P3AT are presented as well as experiments with a larger number of simulated robots which are also presented in [12]. For the two Pioneer P3ATs the graph strategy was converted into paths that are executed in a coordinated fashion between the two robots. Fig. 4 shows the environment used.



(a) (b)

Fig. 4. (a) Map of part of the UCM Science and Engineering building created with a SICK laser. (b) Paths the robots follow to execute a Graph-Clear strategy.

Strategies can inform the movement of robots in a number of ways. Since Graph-Clear strategies are essentially sequences of vertices one first has to move the robots towards the vertex that is being swept and prior to the sweep set up the blocks on neighboring edges. Depending on the number of robots available, one may, usually at earlier steps of the strategy, also sweep multiple vertices at once. At least at one step, however, when the maximum number of robots is required only one vertex can be swept concurrently. For the Line-Clear problem introduced in [5], [11], [12] a Graph-Clear strategy is converted into a schedule for moving sweep lines which are an abstract sensor model that be implemented with a team of robots at a certain cost. For this a number of



(a) (b)

Fig. 2. (a) Map of part of the UCM Science and Engineering. (b) Extracted surveillance graph.

robots simply follow the lines so that their joint sensors at all times cover the entire line.

III. Future directions

In light of the many exciting findings we encountered in this research, we have identified a few interesting research directions that in our opinion deserve more attention in the future. We shall first outline a few precise directions and then more general areas. For Graph-Clear a number of open questions persist, although for some of them we have gathered preliminary evidence suggesting concrete research directions:

- optimal non-contiguous strategies on graphs (probably pseudo-polynomial);
- probabilistically complete anytime algorithm for contiguous strategies on graphs;
- extension of probabilistic sensing model from Graph-Clear to edge-searching;
- NP-hardness of Graph-Clear on planar graphs (Conjecture: yes).

Most of these are rather technical and strictly related to our previous findings for Graph-Clear. A more general paradigm shift concerns the optimization being performed. In all out studies we have targeted the problem of minimizing the number of robots needed to implement a solving strategy (and possibly also to maximize the detection probability in the probabilistic version). Rather different results can be anticipated when one aims to optimize something different, like for example distance travelled (or energy), time to completion, and so on.

For the more general problem of pursuit evasion and search, we instead put forward the following general issues:

- **Realizing there is a third dimension**. Most scholar publications have considered situations where pursuers and evaders move in planar environments, while only limited research has considered situations where motion in three dimensions is allowed. Following the developments observed in numerous other sectors of robotic research, this seems to be an unavoidable evolution.
- **The other side matters**. Pursuit-evasion has been almost invariably intended as the problem of coordinating the motion of pursuers whereas intruders have been often assumed to move randomly, stationary, or moving accordingly to unknown patterns. The other side of the game is no less interesting, but it has been scarcely addressed [1], [14], [15] Not only there are many situations where it may be valuable to remain covert during the mission (wildlife observation, behind the lines operations, etc), but this may also shed some light into the intrinsic limitations, or strengths, of pursuit strategies. Moreover, *smart* evasion strategies may be adopted as benchmark problems to test the effectiveness of pursuit strategies (see next point).
- **Comparing apples with apples**. One of the reasons behind the proliferation of numerous slightly different formalisms for pursuit evasion is also the lack of broadly accepted benchmark problems. This shortcoming is surely common in many robotic applications, but may probably be easier to overcome for problems involving search and pursuit. The availability of benchmark problems will trigger the possibility to compare different solutions not only on a analytical basis, but also numerically.
- **Correctness by design**. Many of pursuit-evasion and search applications are *vital* in the literal sense of the word. For a given pursuit strategy, designing a distributed robot controller that satisfies the specifications by design is of paramount importance. A fertile cross-fertilization with hybrid systems theory seems to be the path to follow, even though only limited attempts have been reported up to now. [17].
- **Interactive strategies**. Strategies are generally global and pre-computed. Changes in the environment or agents that participate in the execution of a search schedule may cause unexpected change. Adapting to

such change will allow the application in dynamic environments and in joint search teams with robots and greatly enhance the potential for applications.

- **Heterogenous teams**. In most scenarios it is unlikely that all robots will have the same capabilities. Even if built the same way they may have defects, different battery levels and so on. Accommodating heterogeneity of teams is hence another crucial, albeit very challenging from a formal perspective, new direction.

IV. Conclusions

We have summarized the current state of the art regarding the Graph-Clear problem and outlined its relationship to other problems relating to the application of pursuit-evasion models to actual robotic search and security applications. A number of technical and practical challenges remain, but we are already equipped to move further towards real robot applications utilizing the insights gained so far.

Acknowledgments

The authors thank the workshop organizers for having taken the lead in creating this stimulating event.

References

[1] T. Bandyopadhyay, Y. Li, M.H. Ang Jr., and D. Hsu. Stealth tracking of an unpredictable target among obstacles. In M. Erdmann, D. Hsu, M. Overmars, and A.F. van der Stappen, editors, *Algorithmic Foundations of Robotics VI*. Springer, 2005.

[2] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Capture of an intruder by mobile agents. In *Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 200–209, New York, NY, USA, 2002. ACM Press.

[3] F. V. Fomin and D. M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399(3):236–245, 2008.

[4] G. Hollinger, A. Kehagias, S. Singh, D. Ferguson, and S. Srinivasa. Anytime guaranteed search using spanning trees. Technical Report CMU-RI-TR-08-36, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2008.

[5] A. Kolling. *Multi-Robot Pursuit-Evasion*. PhD thesis, University of California, Merced, December 2009.

[6] A. Kolling and S. Carpin. The GRAPH-CLEAR problem: definition, theoretical properties and its connections to multirobot aided surveillance. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1003–1008, 2007.

[7] A. Kolling and S. Carpin. Extracting surveillance graphs from robot maps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2323–2328, 2008.

[8] A. Kolling and S. Carpin. Multi-robot surveillance: an improved algorithm for the Graph-Clear problem. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2360–2365, 2008.

[9] A. Kolling and S. Carpin. On weighted edge-searching. Technical Report 01, School of Engineering, University of California, Merced, 2009.

[10] A. Kolling and S. Carpin. Probabilistic Graph-Clear. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3508–3514, 2009.

[11] A. Kolling and S. Carpin. Surveillance strategies for target detection with sweep lines. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5821–5827, 2009.

[12] A. Kolling and S. Carpin. Multi-robot pursuit-evasion without maps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2010. accepted for publication.

[13] A. Kolling and S. Carpin. Pursuit-evasion on trees by robot teams. *IEEE Transactions on Robotics*, 26(1):32–47, 2010.

[14] S. Markov and S. Carpin. A cooperative distributed approach to target motion control in multirobot observation of multiple targets. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 931–936, 2007.

[15] M.S. Marzouqi and R.A. Jarvis. Enhancing self covertness in a hostile environment from expected observers at unknown locations. In *Proceedings of the 9th International Conference on Intelligent Autonomous Systems*, pages 189–196, 2006.

[16] S. Sachs, S. Rajko, and S. M. LaValle. Visibility-based pursuit-evasion in an unknown planar environment. *The International Journal of Robotics Research*, 23(1):3–26, January 2004.

[17] P. Tabuada. *Verification and control of hybrid systems*. Springer, 2009.