ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-1579

Strategic Path Planning Allowing On-the-Fly Updates

Ofri Keidar and **Noa Agmon**[1]

Abstract. This work deals with the problem of strategic path planning while avoiding detection by a mobile adversary. In this problem, an evading agent is placed on a graph, where one or more nodes are defined as *safehouses*. The agent's goal is to find a path from its current location to a safehouse, while minimizing the probability of meeting a mobile adversarial agent at a node along its path (i.e., being captured). We examine several models of this problem, where each one has different assumptions on what the agents know about their opponent, all using a framework for computing node utility. We use several risk attitudes for computing the utility values, whose impact on the actual performance of the path planning algorithms is highlighted by an empirical analysis. Furthermore, we allow the agents to use information gained along their movement, in order to efficiently update their motion strategies on-the-fly. Analytic and empiric analysis show that on-the-fly updates increase the probability that our agent reaches its destination safely.

1 Introduction

The problem of path planning is one of the fundamental problems in the field of agents and robotics [5,7,9,10]. The goal in path planning is to find a sequence of world locations which allows the agent to arrive at its destination while optimizing some criteria, usually minimizing travel cost while avoiding obstacles.

In this paper we introduce a new variant of traditional path planning problem: **Stra**tegic **P**ath **P**lanning (or STRAPP, in short). In this problem, we aim at planning a path for our agent (denoted as R), while avoiding being captured by a mobile adversarial agent (denoted as C). The agents travel about a graph, representing a map of the environment (referred to as *map graph*), where a set of nodes in this graph are defined as *safehouses*. The goal of R is to arrive at one of the safehouses without being intercepted by C on its way there (*capture* it). This problem is applicable in various domains, e.g., delivering humanitarian aid into a hostile area, evading enemy forces in battlefield and even modeling better video games agents.

The problem of traveling in an environment while avoiding threats has been studied from different perspectives [2,4,6,11]. In the problem of Pursuit-Evasion [1,3,8], two rival agents move around the environment (e.g., along the edges of a graph), until the pursuer moves to the evader's location. Most research in pursuit evasion focuses on aspects concerning topology of the graph, for example, on defining properties related to graph theory of the given graph, in order to characterize graphs where the pursuer is guaranteed to capture the evader or finding minimal number of pursuers required. In our problem, however, the evader R moves to a certain destination, and does not try only evade its pursuer. Furthermore, our problem addresses strategic behavior and game theory concepts. R's path is planned based on strategies that take into account its risk attitude.

[1] Bar Ilan University, Israel, email: ofri.keidar@biu.ac.il, agmon@cs.biu.ac.il

We formally introduce the STRAPP problem, and examine different variants of it. We present a framework for computing utilities associated with each node in the graph, computed in polynomial time. This framework is used for finding solutions to the STRAPP problem in all examined variants, which differ in the level of knowledge the agents have on their opponents, and on the risk attitude they adopt.

2 STRAPP Problem Definition

The STRAPP (**Stra**tegic **P**ath **P**lanning) problem is formally defined as follows:
Given a graph $G = (V, E)$, representing a map of the environment (referred to as *map graph*), $V_G \subseteq V$ a set of goal nodes (*safehouses*) and two distinct initial positions of an agent R and an adversarial agent C, find a strategy that will maximize R's chances of reaching some node $v_g \in V_G$ without being captured by C. R is captured by C if both agents reside the same node. R wins if it reached a goal node $v_g \in V_G$ without being captured, while C wins if it captures R.

Note that the strategy may be deterministic or stochastic, and changes based on the knowledge the agents have on their opponent's strategy and location, and on the risk attitude adopted by the agents.

3 Estimating Safety of Map Graph Nodes

We define a *utility* value for each map graph node $v \in V$. The utility of $v \in V$ expresses how *safe* it is for R if moves to v, i.e., how probable it is to evade capture and reach a goal node (i.e., win). This value is derived by evaluating the game configurations (i.e., game states) where R resides at v.

More specifically, a game configuration holds current location of both R,C. The *configuration graph* $G_{conf} = (V_{conf}, E_{conf})$ is defined with a node for each configuration and an edge between any pair of consecutive game states. Configurations matching game states where R wins (*win configuration*) are given a utility of 1, while those matching game states where R loses (*lose configurations*) are given a utility value of 0. For all other configurations, the utility depends on that of its neighbors, and this value is propagated from the *terminal configurations* (i.e., win and lose configurations):
Configurations are traversed along G_{conf} in ascending order from any terminal configuration. The utility value of a configuration $\mathcal{V}$ is computed based only on its neighbors whose utility value had already been computed. Various risk attitudes can be used: risk averse (utility of $\mathcal{V}$ is minimal utility among visited neighbors), risk neutral (average utility among neighbors) or risk seeking (maximal utility among neighbors).

Once all configurations have been given a utility value, the utility for a map graph node $v \in V$ is the average utility of all configurations $\mathcal{V} \in V_{conf}$, such that $\mathcal{V} =< v, u >, u \in V$. This manner

relates to a risk neutral type of player. Other risk attitudes can be applied, e.g., risk averse (utility of a node v is the minimal utility of a configuration $\mathcal{V} =< v, u >$) or risk seeking (utility of v is the maximal one of a configuration $\mathcal{V} =< v, u >$).

4 Constructing Stochastic Motion Strategies

We assume that the adversarial agent C is capable of performing the same computations as we do. If our agent R would have followed a known deterministic movement policy, then given its initial location, C is capable of knowing its current location at each turn. Hence, there are cases where R cannot win in such model. However, if both players choose their next move according to a stochastic strategy, even if these strategies are known for each player, then there is a non-zero probability of R choosing a move that results in safe arrival to a goal node $v_g \in V_G$. As a matter of fact, in this case C benefits also from a stochastic motion pattern. A stochastic strategy P_v (i.e., a probability distribution over possible actions) is associated for each node $v \in V$ ($P_v[u]$ is the transition probability from v to u, $(v,u) \in E$), in order to employ uncertainty among the opponent.

Given the utility value of each map graph node, $P_v[u]$ ($(v,u) \in E$) is u's utility value normalized by utility values of all v's neighbors. This implies that R has a greater chance to move towards *safer* nodes, i.e., nodes with greater utility. Moving towards the neighboring node with the maximal utility value makes R's moves predictable for C, which can compute these utility values (C is assumed to have equal computational capabilities). However, such stochastic motion leads to uncertainty regarding R's assumed path.

We have proven that if C follows these stochastic strategies, as well, then an equilibrium is achieved.

5 Strategy Updates On-the-Fly

So far, strategies were computed offline, such that they aim to reduce the probability that C captures R, based on the map graph's topology. However, relying solely on graph topology, i.e., offline planning, means no reaction to new information gained while moving around the map graph. If some nodes can be observed by the other nodes (which can be considered as *viewpoints*), these nodes may provide information (or, *knowledge*) regarding an agent, e.g., tracks the agent had left behind or perhaps whether the agent currently resides at the node. When the agents follow the strategies computed as stated in Section 4, they can use these viewpoints in order to acquire information concerning their opponent's location or visited nodes, and update their strategies accordingly (each agent and its own objective). The updates are extremely efficient, i.e., linear in the number of neighbors.

The contribution of these runtime updates had been proven both theoretically and empirically.

6 Empirical evaluation

We have evaluated our path planning strategies, using different risk attitudes for computing the utility functions, and examined the effect of online strategies update. Utility functions were used in order to compute a utility value for a map graph node or a configuration node (given the values of the node's neighbors) and also used to evaluate the information obtained at a node visited by an agent.

A collection of graphs with 10 to 60 (with jumps of 5) nodes was randomly generated (40 of each size), so were the visibility edges. For each number of nodes, 10% of the nodes where randomly set as goal nodes (i.e., safehouses).

An experiment was executed for each combination of node, configuration and information utility functions for R,C. Each experiment was repeated 20 times for each graph within the graphs collection. Each time new starting locations for both agents were randomly chosen (not among the safehouses). For each graph size, combination of node, configuration and information utilities, the average winning rate of R was calculated (R*'s winning rate*).

Figure 1 shows the results when both agents update their strategies on-the-fly. R was tested with several behaviors for on-the-fly strategy updates. The curve labeled as OFFLINE is where R did not perform any on-the-fly updates (i.e., runs offline). In order to specifically examine the influence of on-the-fly updates on R's winning rate, when R did update its strategies on-the-fly, executions where R did not observe C even once were discarded.

ANOVA test with $\alpha = 0.05$, followed by post hoc test, has confirmed that R's winning rate is significantly increased with on-the-fly updates.

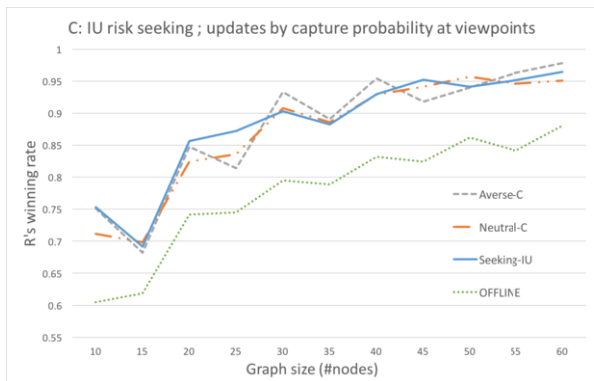

Figure 1: R*'s winning rate when both* R,C *update their strategies on-the-fly.* C*'s information utility is risk seeking, at a viewpoint* C *updates by capture probability*

REFERENCES

[1] Brian Alspach, 'Searching and sweeping graphs: a brief survey', *Le matematiche*, **59**(1, 2), 5–37, (2006).
[2] Richard B Borie, Craig A Tovey, and Sven Koenig, 'Algorithms and complexity results for pursuit-evasion problems.', in *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, volume 9, pp. 59–66, (2009).
[3] Timothy H Chung, Geoffrey A Hollinger, and Volkan Isler, 'Search and pursuit-evasion in mobile robotics', *Autonomous Robots*, **31**(4), 299–316, (2011).
[4] Fedor V Fomin and Dimitrios M Thilikos, 'An annotated bibliography on guaranteed graph searching', *Theoretical Computer Science*, **399**(3), 236–245, (2008).
[5] Jean Claude Latombe, *Robot Motion Planning*, Boston: Kluwer Academic Publishers, 1991.
[6] Mohamed Marzouqi and Ray Jarvis, 'Covert path planning for autonomous robot navigation in known environments', in *Proc. Australasian Conference on Robotics and Automation, Brisbane*. Citeseer, (2003).
[7] Andreas C Nearchou, 'Path planning of a mobile robot using genetic heuristics', *Robotica*, **16**(05), 575–588, (1998).
[8] Richard Nowakowski and Peter Winkler, 'Vertex-to-vertex pursuit in a graph', *Discrete Mathematics*, **43**(2), 235–239, (1983).
[9] Clement Petres, Yan Pailhas, Pedro Patron, Yvan Petillot, Jonathan Evans, and David Lane, 'Path planning for autonomous underwater vehicles', *IEEE Transactions on Robotics*, **23**(2), 331–341, (2007).
[10] Anthony Stentz, 'Optimal and efficient path planning for partially-known environments', in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 3310–3317, (1994).
[11] Roi Yehoshua and Noa Agmon, 'Adversarial modeling in the robotic coverage problem', in *Proceedings of International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, (2015).