Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)

Finding Centroids and Minimum Covering States in Planning

Alberto Pozanco, Yolanda E-Martín, Susana Fernández, Daniel Borrajo
Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30. 28911 Leganés (Madrid), Spain
apozanco@pa.uc3m.es, yescuder@inf.uc3m.es, sfarregu@inf.uc3m.es, dborrajo@ia.uc3m.es

Abstract

In automated planning, the most common task consists of finding a plan that achieves a set of goals. In this paper, we focus on a different task; that of finding states that minimize some goal-related metric. First, we present some domains for which that task is useful. Second, we propose two of such types of states: (1) centroid states, which minimize the distance to all the goals in the problem; and (2) minimum covering states, which minimize the maximum distance to any of the goals. Third, we define optimal and suboptimal algorithms to find such states. Finally, we show some experimental results in planning instances from different domains.

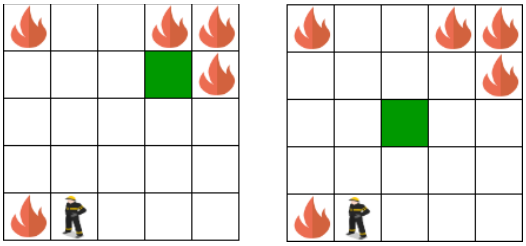

Figure 1: The green square represents the centroid (left image) and minimum covering state (right image) of the planning instance.

Introduction

Consider a domain like the one shown in Figure 1 where a forest ranger has to put out fires. Suppose that some fires might arrive dynamically in the locations marked with a flame. Under these circumstances, the ranger should generate a plan to set the camp at a location that minimizes the cost (time) of a plan to put out any fire that might arrive.

In this short paper, we propose to find two types of states in planning instances, which allow agents to reason about the potential goals in two different ways. The first type of states are the *centroids*, which are those states that minimize the average of the costs towards all goals in the problem. In other words, agents would approach to most of the potential goals. These states have a direct impact in anticipatory planning (Burns et al. 2012; Pozanco, Fernández, and Borrajo 2018; Fuentetaja, Borrajo, and de la Rosa 2018), which focuses on generating plans prior to the arrival of goals. Examples of domains in which finding centroid states can be useful are: locations where the ranger should set a camp to put out potential fires; areas to be patrolled by police (or robots), where potential incidents might happen; source locations for picking up customers by taxis or packages by delivery companies; or locations where some natural disaster might happen and drones (rovers or satellites) are needed to take images or any other kind of intervention action.

The second type of states are *minimum covering* states, which are those states that minimize the maximum distance to any of the goals. In other words, agents would not be far from any potential goal. These states have a direct impact in competitive domains. For example, in scenarios where an agent tries to hide her goal to an opponent as long as it can (Keren, Gal, and Karpas 2016; Masters and Sardina 2017; Kulkarni, Srivastava, and Kambhampati 2018), or wants to block an enemy (Speicher et al. 2018; Pozanco et al. 2018). In the former case, plans that move towards minimum covering states make the enemy's goal recognition task more difficult because the agent moving to that state is not fully compromising to any goal. In the latter case, intelligent agents should move towards minimum covering states to increase the chances of stopping the opponent, regardless of the goal she ends up trying to achieve. Following our example, if the ranger sets the camp at a minimum covering state: (1) it would be harder for a possible arsonist to know the fire area the ranger is trying to cover; and (2) the ranger would not be far from any potential fire.

Related work solves a similar subtask when obfuscating goals (Keren, Gal, and Karpas 2016; Kulkarni, Srivastava, and Kambhampati 2018). However, they do not compute centroids nor minimum covering states, nor they consider the full set of potential goals.

The main contributions of this paper are: (1) definition of two new planning tasks (computing the sets of centroids and minimum covering states of a set of goals); and (2) definition of an algorithm for computing those two sets optimally and suboptimally. We present an empirical study that shows the quantitative performance of our approach, as well as shedding some light on the kind of states that can be obtained.

Background

Classical automated planning is the task of choosing a sequence of actions such that, when applied in a given initial state, it results in a goal state. Formally, a STRIPS planning task can be defined as a tuple $\Pi = \langle F, A, I, G \rangle$, where F is a set of propositions, A is a set of instantiated actions, $I \subseteq F$ is an initial state, and $G \subseteq F$ is a set of goals. Each action $a \in A$ is described by a set of preconditions (pre(a)), which represent literals that must be true in a state to execute an action, and a set of effects (eff(a)), which are the literals that are added (add(a) effects) or removed (del(a) effects) from the state after the action execution. The definition of each action might also include a cost $c(a)$ (the default cost is one). The execution of an action a in a state s is defined by a function γ such that $\gamma(s, a) = (s \setminus \text{del}(a)) \cup \text{add}(a)$ if pre(a)$\subseteq s$, and s otherwise (it cannot be applied). The output of a planning task is a sequence of actions, called a plan, $\pi = (a_1, \ldots, a_n)$. The execution of a plan π in a state s can be defined as:

$$\Gamma(s, \pi) = \begin{cases} \Gamma(\gamma(s, a_1), (a_2, \ldots, a_n)) & \text{if } \pi \neq \emptyset \\ s & \text{if } \pi = \emptyset \end{cases}$$

A plan π is valid for solving Π iff $G \subseteq \Gamma(I, \pi)$. The plan cost is commonly defined as $c(\pi) = \sum_{a_i \in \pi} c(a_i)$.

In this work, we solve the following alternative planning task $P = \langle F, A, I, \mathcal{G} \rangle$. Given the same F, A and I as Π; and a set of potential goal sets $\mathcal{G}$; find a set of states $\mathcal{S}$ that are either (a) the centroids or (b) the minimum covering states of P. In the next section, we formalize both types of states.

Planning Centroids and Planning Minimum Covering States

In a geometric space, centroid points are those that minimize the distance to all the points in a given set. Likewise, minimum covering points are those that are in the center of a minimum covering sphere, i.e., the smallest sphere that contains all of a given set of points (Sylvester 1857). This point minimizes the maximum distance to any point in the set. Computing these points in a geometric space is straightforward since: (1) every point is reachable from any other; (2) point's features (coordinates) are fully specified; and (3) the distance from one point to another can be simply calculated as the euclidean distance of its coordinates.

In automated planning, there are states instead of points. These states: (1) are not necessarily reachable from all other states; (2) are not necessarily fully specified (we can find partial states); and (3) the computation of the distance between two states becomes more complicated. Therefore, next we define reachability and distance between states for automated planning.

Definition 1 (Reachable State) *A state s is reachable from I (or simply reachable) iff there is at least one valid plan π such that its execution from I, $\Gamma(I, \pi)$, achieves s; i.e. $s \subseteq \Gamma(I, \pi)$. We will refer to the set of all reachable states from I in P as $\mathcal{R}_P$.*

Next, we need to define a distance between states. Since goals are partial states, we cannot easily compute the distance between two goals. Instead, we can compute the distance between a regular (full) state and a partial state (goal). The set of distances $\mathcal{D}$ from a fully specified state s to all goals in $\mathcal{G}$ given P can be defined as:

$$\mathcal{D}_s(P) = \{(G_i, d) \| G_i \in \mathcal{G}, d = c(F, A, s, G_i, h)\}$$

We refer to this procedure as COMPUTEDISTANCES, where $c(F, A, s, \mathcal{G}, h)$ represents the cost of achieving each set of propositions $G_i \in \mathcal{G}$ from s using h as the function that computes the cost. This function can compute either the actual optimal cost h^* of achieving G_i from s, or an estimation of that cost (as the one returned by a heuristic function).

We can now define centroid and minimum covering states using the arithmetic mean, μ, and the maximum distance over a set, $\max$, as follows:

Definition 2 (Planning Centroid State) *A state $s \in \mathcal{R}_P$ is a centroid state of a planning task P iff $\forall s' \in \mathcal{R}_P$, $\mu_d(\mathcal{D}_s(P)) \leq \mu_d(\mathcal{D}_{s'}(P))$ and $\mathcal{D}$ is computed using h^*. The set of centroid states of a planning task P is denoted as $\mathcal{C}^*(P)$.*

Definition 3 (Planning Minimum Covering State) *A state $s \in \mathcal{R}_P$ is a minimum covering state of a planning task P iff $\forall s' \in \mathcal{R}_P$, $\max_d(\mathcal{D}_s(P)) \leq \max_d(\mathcal{D}_{s'}(P))$ and $\mathcal{D}$ is computed using h^*. The set of minimum covering states of a planning task P is denoted as $\mathcal{M}^*(P)$.*

Computing Centroids and Minimum Covering States in Planning

We propose C&MCS, a common algorithm that can compute centroids and minimum covering states both optimally and suboptimally, depending on the input parameters. Algorithm 1 details the full procedure. It consists of a Best-First Search algorithm that receives as input: a planning task P, the function h used to compute $\mathcal{D}$, the evaluation function used to sort the nodes in the open list e, and a parameter o that determines whether the full state space reachable from the initial space $\mathcal{R}_P$ should be explored or not.

The function COMPUTEHEURISTICS returns two heuristic values: the arithmetic mean μ and the maximum value $\max$ of the set of costs in $\mathcal{D}_s$, which are computed using h. The states are compared and sorted in the open list according to the evaluation function e. When looking for centroid states, the evaluation function minimizes the arithmetic mean $\mu(\mathcal{D}_s)$, breaking ties in favour of the minimum $\max(\mathcal{D}_s)$. When looking for minimum covering states, the evaluation function minimizes $\max(\mathcal{D}_s)$, breaking ties in favor of the minimum arithmetic mean $\mu(\mathcal{D}_s)$. The algorithm returns $\mathcal{S}$, which corresponds to the set of states $s \in \mathcal{S}$ that minimize the evaluation function e. We are not taking into account the cost of achieving these states from the initial state, since centroids and minimum covering states, apart form being reachable, are only related to the goals.

To compute optimal centroids $\mathcal{C}^*$ and minimum covering states $\mathcal{M}^*$ for planning, we need to know the value of h^* from every state $s \in \mathcal{R}_P$ to every set of goals $G_i \in \mathcal{G}$. In classical forward search planning, h^* is computed generating an optimal plan from a state to the goal. This means that, $\mathcal{C}^*$ and $\mathcal{M}^*$ will compute n optimal plans $\forall s \in \mathcal{R}_P$, where $n = |\mathcal{G}|$. On the other hand, if we are fine with suboptimal solutions, (1) we can use any cost estimator h to compute

Algorithm 1 C&MCS

Inputs: $P = \langle F, A, I, \mathcal{G} \rangle$, h, e, o
Outputs: $\mathcal{S}$

$\mathcal{D}_\mathcal{S} \leftarrow$ComputeDistances$(F, A, I, \mathcal{G}, h)$
$h_\mathcal{S} \leftarrow$ ComputeHeuristics$(\mathcal{D}_\mathcal{S})$
open $\leftarrow \{\langle I, h_\mathcal{S} \rangle\}$
$\mathcal{S} \leftarrow \{\langle I, h_\mathcal{S} \rangle\}$
while open $\neq \emptyset$ **do**
  $n, h_n \leftarrow$ pop(open, e)
  if $e(h_n) > e(h_\mathcal{S})$ **and** not o **then**
    return $\mathcal{S}$
  successors $\leftarrow$ GenerateSuccessors(n)
  for s **in** successors **do**
    $\mathcal{D}_s \leftarrow$ComputeDistances$(F, A, s, \mathcal{G}, h)$
    $h_s \leftarrow$ ComputeHeuristics$(\mathcal{D}_s)$
    if $e(h_s) < e(h_\mathcal{S})$ **then**
      $h_\mathcal{S} \leftarrow h_s$
      $\mathcal{S} \leftarrow \{\langle s, h_\mathcal{S} \rangle\}$
    else if $e(s) = e(\mathcal{S})$ **then**
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{\langle s, h_s \rangle\}$
    open $\leftarrow$ open$\cup\{\langle s, h_s \rangle\}$
return $\mathcal{S}$

the distance to the goals; and (2) we can decide whether to explore the entire state space or not (parameter o). Suboptimal solutions that explore the full state space are denoted as $\mathcal{C}^f$ and $\mathcal{M}^f$. Suboptimal solutions that greedily stop the algorithm when there is no state in the open list with a better heuristic value than the best states visited so far are denoted as $\mathcal{C}$ and $\mathcal{M}$.

Experimental Evaluation

We implemented C&MCS in FastDownward (Helmert 2006). We vary the parameters h, e, and o of C&MCS to get the six different algorithms shown in Table 1. We use the FF heuristic (Hoffmann and Nebel 2001) to compute $\mathcal{D}$ in the suboptimal algorithms. To compute h^* in the optimal algorithm, we run n optimal plans at each search node using A^* with the LMCUT heuristic (Helmert and Domshlak 2009), where $n = |\mathcal{G}|$. The experiments were run on Intel(R) Xeon(R) CPU X3470 @ 2.93GHz machines with a time limit of 3600s and a memory limit of 16GB. We consider

Version	h	e	o
$\mathcal{C}^*$	h^*	$< \mu(\mathcal{D}), \max(\mathcal{D}) >$	Yes
$\mathcal{C}^f$	FF	$< \mu(\mathcal{D}), \max(\mathcal{D}) >$	Yes
$\mathcal{C}$	FF	$< \mu(\mathcal{D}), \max(\mathcal{D}) >$	No
$\mathcal{M}^*$	h^*	$< \max(\mathcal{D}), \mu(\mathcal{D}) >$	Yes
$\mathcal{M}^f$	FF	$< \max(\mathcal{D}), \mu(\mathcal{D}) >$	Yes
$\mathcal{M}$	FF	$< \max(\mathcal{D}), \mu(\mathcal{D}) >$	No

Table 1: C&MCS versions.

two types of domains: with and without conflicting goals. In domains without conflicting goals, there is always a state $s \in \mathcal{R}_P$ where all $G_i \in \mathcal{G}$ can be achieved at the same time. Such state s has $\mu(D_s) = \max(D_s) = 0$. Therefore, it is both a centroid and a minimum covering state. This is the case of all solvable instances of IPC domains, where each $G_i = \{g_j\}$ where the g_j are the goals of the standard planning task. For instance, in the Barman domain, centroids and minimum covering states are any state where all shots have been generated. When we run our algorithms in these domains, we obtain those states.

In the second type of domains, $\mathcal{G}$ consists of conflicting goals. This means that there is no state $s \in \mathcal{R}_p$ where all $G_i \in \mathcal{G}$ can be jointly achieved. This is the case of domains described in the Introduction. We selected two domains with conflicting goals that encompass many others.

Ranger, the domain of the example presented in the Introduction, where a ranger has to set a camp at a location that minimizes the cost of a plan that puts out one of the fires that may arrive in the near future. We generated 20 random problems in 20×20 maps. A 15% of the cells can be obstacles that the ranger cannot pass through. For each problem, we randomly generated 4 potential goals and the initial state. This domain represents the set of domains where an agent has to move to a given location such that it minimizes time to achieve goals once they appear. Examples of such domains are patrolling, surveillance, or service providing domains.

Blocks Words (Ramírez and Geffner 2009), a variation of the well-known Blocksworld domain. Here, each block contains a letter and an agent has to build a word using the different letters. We generated 20 random problems with the same 5 blocks with a random initial distribution. Each problem has 3 potential goals, randomly selected from a dictionary containing 26 English words that can be built using the 5 blocks. The length of these words is between three and five letters. An example of a set $\mathcal{G}$ may be to build `APE`, `EAR`, or `PEAR`, where for the first word, the agent would need to make true the propositions `(clear A)`,`(on A P)`,`(on P E)`, and `(ontable E)`. This domain can be seen as a competitive scenario where an agent hides the real word that she is building to an opponent agent. This example subsumes many other competitive and privacy protection domains such as cybersecurity and games.

Quantitative Evaluation

Table 2 summarizes the results of a quantitative evaluation. For each domain, each row shows the results obtained by each of the six versions. Each column represents different measures of quality and performance over the 20 problems. (1) μ: arithmetic mean of $\mathcal{D}_s$, for any $s \in \mathcal{S}$; (2) $\Delta(\mu)$: percentage of improvement of μ for any returned state $s \in \mathcal{S}$ with respect to the initial state I; (3) $\max$: maximum value in $\mathcal{D}_s$, for any $s \in \mathcal{S}$; (4) $\Delta(\max)$: percentage of improvement of $\max$ for any returned state $s \in \mathcal{S}$ with respect to the initial state I; and (5) t: time in seconds to compute $\mathcal{S}$. In addition, we compute the suboptimality ratio as the μ or $\max$ values returned by the suboptimal versions divided by the μ or $\max$ values returned by the optimal version.

We get the best results for μ when computing centroid states and the best $\max$ results when computing minimum covering states in all the domains. In general, all versions compute centroids and minimum covering states that improve μ and $\max$ respecting the initial state. However, this

improvement varies among instances due to: (1) the given initial state, which may be close/far from the computed centroid or minimum covering state; and (2) the heuristic, which can overestimate or underestimate the actual cost.

The execution times of $\mathcal{C}^*$ and $\mathcal{M}^*$ are three orders of magnitude higher than the ones of the suboptimal versions. They need to explore the whole state space reachable from the initial state, computing an optimal plan to each goal in every state. Suboptimal versions present a good balance between quality and time, computing average/max of distances that are closer to the optimal ones in less than a second. We also observe this behavior in Figure 2, where we measure the suboptimality ratios obtained in both domains. The upper image shows the degradation of $\mathcal{C}^f$ and $\mathcal{C}$ with respect to the μ values returned by the optimal algorithm, $\mathcal{C}^*$. $\mathcal{C}^f$ is always able to return states that are at most 1.6 times suboptimal and whose median is less than 1.3. The $\mathcal{C}$ version returns worse solutions, as shown by the bigger variety of suboptimality values. However, its median value is 1.5.

Domain	Version	μ	$\Delta(\mu)$	max	Δ(max)	t
RANGER	$\mathcal{C}^*$	**8.0**±2	**41.6**±22	11.8±3	41.1±18	530.6±7
	$\mathcal{C}^f$	**8.0**±2	**41.6**±22	11.8±3	41.1±18	0.2±0
	$\mathcal{C}$	8.4±2	38.9±21	11.9±3	40.8±17	0.1±0
	$\mathcal{M}^*$	8.5±2	38.1±21	**10.6**±2	**46.7**±17	530.6±7
	$\mathcal{M}^f$	8.5±2	38.1±21	**10.6**±2	**46.7**±17	0.2±0
	$\mathcal{M}$	9.2±3	33.9±21	11.4±3	43.2±17	0.1±0
BLOCKS WORDS	$\mathcal{C}^*$	**6.5**±1	**35.9**±16	9.4±1	26.3±16	358.3±4
	$\mathcal{C}^f$	8.1±2	21.1±17	12.6±3	2.9±20	0.1±0
	$\mathcal{C}$	10.1±3	5.8±9	12.5±3	5.6±8	0.1±0
	$\mathcal{M}^*$	7.0±1	30.9±18	**7.4**±1	**42.4**±10	358.3±4
	$\mathcal{M}^f$	8.5±2	16.6±24	11.2±2	12.0±23	0.1±0
	$\mathcal{M}$	10.1±3	5.8±9	12.5±3	5.6±8	0.1±0

Table 2: Comparison among different versions. Numbers represent averages and standard deviations over the set of problems.

In domains where the FF heuristic computes more accurate estimations, as in the case of Ranger, the performance of suboptimal versions is closer to the optimal one. As a matter of fact, FF is a perfect estimator in that domain; the $\mathcal{M}^f$ and $\mathcal{C}^f$ versions get the same results as their respective optimal versions in all the problems, since both are exploring all $\mathcal{R}_P$. The $\mathcal{M}$ and $\mathcal{C}$ versions get similar results in Ranger, but they can fall in local minima. To test their accuracy in the presence of different local minima, we generated problems with increasing percentages of obstacles in Ranger. Results are depicted in the lower image of Figure 2. In the absence of obstacles, the $\mathcal{M}$ version is able to get optimal minimum covering states. As the percentage of obstacles increases, some solutions start moving away from the optimum. However, the performance of this version is equivalent to the optimal one on average.

The scalability of the algorithm depends on $|\mathcal{R}_p| \times |\mathcal{G}|$. We performed a test where $|\mathcal{G}|$ increases in powers of 2 from 2 to 64. Results show that the optimal version did not solve the problems within the time bounds when $|\mathcal{G}| > 16$. The suboptimal versions solved all problems in less than a second with a linear increase in solving time.

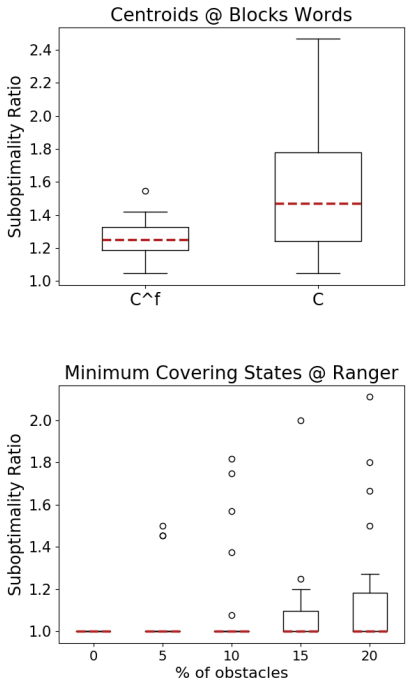

Figure 2: The upper image shows the degradation of $\mathcal{C}^f$ and $\mathcal{C}$ with respect to $\mathcal{C}^*$ in Blocks Words. The lower image shows the degradation of $\mathcal{M}$ with respect to $\mathcal{M}^*$ when increasing the obstacles in Ranger.

Qualitative Evaluation

It is easy to visualize centroids and minimum covering states in path-planning domains such as Ranger. However, we might ask how they look like in other domains. Figure 3 shows a graphical representation of $\mathcal{M}^*$, $\mathcal{M}^f$, $\mathcal{C}^*$, and $\mathcal{C}^f$ in two different Blocks Words problems. The first example depicts minimum covering states for the set of potential goals $\mathcal{G}$ in the cloud. In the solutions given by $\mathcal{M}^*$ and $\mathcal{M}^f$, the agent is not fully committed to build any of the goals. In other words, the agent would not be far from building any $G_i \in \mathcal{G}$. However, in the solution given by $\mathcal{M}^*$, it would be less costly to build any of the potential words than using $\mathcal{M}^f$. The second example depicts centroid states in a different problem. In the solutions given by $\mathcal{C}^*$ and $\mathcal{C}^f$, the agent could build most of the $G_i \in \mathcal{G}$ with very little cost, since each solution already contains a word of $\mathcal{G}$. However, in the solution given by $\mathcal{C}^*$, it would be a bit less costly to build the word `PEA`.

Discussion and Future Work

We introduced a novel planning task, which consists of computing the set of centroids and minimum covering states of a set of potential goals. This highlights how finding such states can be very useful in many applications, which range from anticipatory planning to competitive environments. Finally, we presented three different versions of C&MCS to compute centroids and minimum covering states.

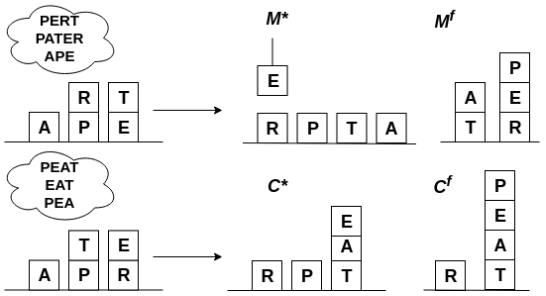

Figure 3: Optimal Centroid and Minimum Covering State in two different instances of the Blocks Words domain.

The behavior of the suboptimal versions is not only related to the presence of local minima but also to the use of accurate heuristics closer to h^*. In this paper, we used the FF heuristic, but in future work we would like to compare algorithm's performance by using others. Including non-uniform goal's probability distribution is straightforward as well as including a cost bound or considering the cost of reaching the states from the initial state when looking for centroids and minimum covering states. We would also like to explore these options in future work. Finally, there might exist more efficient ways of computing these states optimally by using backward and/or symbolic search. It would be interesting to see how to tackle this problem from that perspective.

Acknowledgements

This work has been partially funded by FEDER/Ministerio de Ciencia, Innovación y Universidades – Agencia Estatal de Investigación/TIN2017-88476-C2-2-R and RTC-2016-5407-4.

References

Burns, E.; Benton, J.; Ruml, W.; Yoon, S. W.; and Do, M. B. 2012. Anticipatory on-line planning. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*.

Fuentetaja, R.; Borrajo, D.; and de la Rosa, T. 2018. Anticipation of goals in automated planning. *AI Commun.* 31(2):117–135.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*.

Helmert, M. 2006. The fast downward planning system. *J. Artif. Intell. Res.* 26:191–246.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res.* 14:253–302.

Keren, S.; Gal, A.; and Karpas, E. 2016. Privacy preserving plans in partially observable environments. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 3170–3176.

Kulkarni, A.; Srivastava, S.; and Kambhampati, S. 2018. A unified framework for planning in adversarial and cooperative environments. In *Proceedings of the 6th Workshop on Planning and Robotics, ICAPS 2018, Delft, Netherlands*.

Masters, P., and Sardina, S. 2017. Deceptive path-planning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 4368–4375.

Pozanco, A.; E-Martín, Y.; Fernández, S.; and Borrajo, D. 2018. Counterplanning using goal recognition and landmarks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, 4808–4814.

Pozanco, A.; Fernández, S.; and Borrajo, D. 2018. Learning-driven goal generation. *AI Commun.* 31(2):137–150.

Ramírez, M., and Geffner, H. 2009. Plan recognition as planning. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, 1778–1783.

Speicher, P.; Steinmetz, M.; Backes, M.; Hoffmann, J.; and Künnemann, R. 2018. Stackelberg planning: Towards effective leader-follower state space search. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Sylvester, J. J. 1857. A question in the geometry of situation. *Quarterly Journal of Pure and Applied Mathematics* 1.