Proceedings of the Thirty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2025)

Strategies to Improve Goal Selection in Satisficing Oversubscription Planning

Angel García-Olaya[1], **Patricia Riddle**[2], **Michael Barley**[2]

[1] Computer Science and Engineering Department, Universidad Carlos III de Madrid, Spain
[2] University of Auckland, New Zealand
agolaya@inf.uc3m.es, p.riddle@auckland.ac.nz, m.barley@auckland.ac.nz

Abstract

Oversubscription planning (OSP) tackles the infeasibility of finding a plan that achieves all goals, due to a limited resource, typically a cost bound. The objective is to discover a plan under this cost bound that maximizes the utility.

A leading-edge technique in satisficing OSP employs relaxed plans to estimate the cost of achieving goals and focuses planning efforts on goals deemed attainable based on this estimation. However, this approach faces two main challenges: the time required to calculate all estimations can result in no effective goal selection, and using relaxed plans often underestimates the real cost, leading to sets of oversubscribed goals.

To address these challenges, our paper studies two solutions: computing the estimations only when needed and using satisficing plans instead of relaxed plans to calculate cost estimations. Experiments show that using satisficing plans to calculate the estimations offer advantages in terms of initial plan utility, but the gap between both approaches narrows when more time is given for plan refinement.

Introduction and Related Work

Oversubscription planning (OSP) (Smith 2004) is a subset of Automated Planning (AP) that confronts scenarios where resources are constrained, impeding the attainment of all goals. Typically, this limitation is depicted through a bound on the plan cost, while goals are assigned utilities. The primary challenge thus becomes devising a plan within the cost bound that maximizes utility. In *satisficing* OSP the objective is to find the best plan in terms of utility in the alloted time, not necessarily the optimal one. OSP shares connections with other AP problems such as COST-BOUNDED or NET-BENEFIT planning (Domshlak and Mirkis 2015).

Three approaches are reported in the literature to tackle OSP: (1) incrementally seeking plans of greater utility considering all the goals (Baier, Bacchus, and McIlraith 2009; Benton, Coles, and Coles 2012; Domshlak and Mirkis 2015; Muller and Karpas 2018; Speck and Katz 2021); (2) converting the problem into a different form (Katz et al. 2019; Katz and Keyder 2022; van den Briel et al. 2004); and (3) performing an upfront selection of goals and focusing just on them (García-Olaya, de la Rosa, and Borrajo 2021; Sanchez-Nigenda and Kambhampati 2005; Smith 2004).

Currently, Relaxed Plan Linear Distances (RPLD) (García-Olaya, de la Rosa, and Borrajo 2021) finds the highest utility plans in satisficing OSP. It performs upfront selection of goals, using relaxed plans to compute cost estimations for individual goals and for pairs of goals. RPLD usually underestimates the actual cost of the set of selected goals, which frequently remains oversubscribed. In such cases, the goals added last are removed until a plan for the remaining ones is found. The authors observe that in 8 out of the 12 domains, over 50% of the initially chosen sets seem to be still oversubscribed. In 4 domains this percentage increases to over 80%. In certain instances, the oversubscription is so pronounced that considerable time is spent removing goals, resulting in the exhaustion of time without finding a plan. This is exacerbated by the fact that all the estimations are calculated beforehand, even if they are not finally used.

A logical progression involves (1) enhancing the accuracy of cost estimations and (2) calculating them only when necessary. While more precise heuristics (Domshlak, Hoffmann, and Katz 2015; Refanidis and Vlahavas 2001) to calculate cost estimations could be contemplated, an even more accurate method entails using satisficing plans. This is typically dismissed as infeasible due to the computational complexity of planning (Bylander 1994), but that generic complexity is reduced in some cases. For example, in many International Planning Competition (IPC) domains, one-goal problems can be solved in polynomial time by a blind algorithm pruning states based on their *novelty*; a measure of their dissimilarity from any other already visited state (Lipovetzky and Geffner 2012).

Employing a methodology akin to that of RPLD, the primary objectives of the paper are:

1. Assess the impact on the utility of lazy cost estimations calculation. This will allow finding solutions where RPLD exceeds the time bound, and will give extra time to find better plans in the remaining problems.
2. Assess the practical viability of novelty-based planners for cost estimations: their polynomial complexity comes at the cost of incompleteness in some domains, so it is uncertain whether the approach is feasible in practice.
3. Examine the impact of the enhanced accuracy of cost estimations on goal selection and plan utility.

The paper is structured as follows. The next section

presents background about OSP and novelty-based planning. Then, the two strategies to improve goal selection and their formal properties are shown. Results of the new approaches are compared to RPLD. Finally, conclusions and future work are presented.

Background

An OSP task is a tuple $P_{OSP} = \{F, A, I, G, c, u, c_b\}$, where: F is a finite set of fluents; A is a finite set of actions, mapping a set of fluents (a state) to another; $I \subseteq F$ is the initial state ; $G \subseteq F$ is the set of goals; $c : A \rightarrow \mathbb{R}_0^+$ is a cost function; $u : 2^F \rightarrow \mathbb{R}_0^+$ is a utility function; and $c_b \in \mathbb{R}^+$ is a cost bound.

The utility function maps a set of fluents to a number, the *utility* of that set. It is typically defined over individual fluents $f \in F$, so $u : F \rightarrow \mathbb{R}_0^+$. Fluents $f \notin G$ with $u(f) > 0$ are termed soft-goals. In contrast, fluents in G are considered hard-goals and commonly assigned zero utility. Most approaches in the literature work with no hard-goals ($G = \emptyset$). We will follow this approach and use the terms *goal* and *soft-goal* interchangeably. Generally, additive utilities are considered (Do et al. 2007), so the utility of a set of fluents is the sum of their utilities. As in (García-Olaya, de la Rosa, and Borrajo 2021) we will consider additive utilities of individual fluents.

Each action $a \in A$ is characterized by a tuple $\{pre(a) \subseteq F, add(a) \subseteq F, del(a) \subseteq F\}$. An action sequence $\pi = (a_1, a_2, ..., a_n)$ is applicable in P_{OSP} if there exists a sequence of states $(s_0, s_1, ..., s_n)$, where $s_0 = I$, and for all $i \in \{1...n\}$, $pre(a_i) \subseteq s_{i-1}$ and $s_i = (s_{i-1} \setminus del(a_i)) \cup add(a_i)$. The cost of π is denoted as $C(\pi) = \sum_{a_i \in \pi} c(a_i)$, where $c(a_i)$ represents the cost of action a_i. An action sequence π, is a plan for P_{OSP} if it is applicable, $G \subseteq s_n$, and $C(\pi) \leq c_b$. The utility of π is $U(\pi) = \sum_{f_i \in s_n} u(f_i)$.

RPLD's selection of goals is based on the concepts of *Distance to a goal from the initial state* (Δ_{Ix}) and *Distance between two goals* (Δ_{xy}). The first one estimates the cost of achieving any goal from the initial state. It is defined as the cost of the relaxed plan achieving goal x. Meanwhile, Δ_{xy} estimates the cost of obtaining y once x has been reached. It is defined as the cost of the relaxed plan achieving y from the end state of the relaxed plan found to compute Δ_{Ix}. A planning task is relaxed by removing the delete effects of actions, so action a is relaxed as $a' = \{pre(a), add(a), \emptyset\}$.

Goals are added from highest to lowest utility, using distances as tie-breaker, while the estimated cost of the set is within the cost bound. Then, an external planner tries to find a solution for the selected set for 90 seconds. If no plan is found, the branch leading to that goal set is pruned, the search backtracks, removing the last added goal, and the external planner is invoked again. The procedure continues until a plan has been found or the set is empty. In that later case, search starts again by exploring the branch of the goal with the second highest utility. Once a plan has been found, the search continues, backtracking as necessary, but the external planner is only invoked if the utility of a terminal node surpasses the best found utility. If the entire search space under the cost bound has been explored, the cost constraint is relaxed and sets whose estimated cost exceeds c_b are considered, planning every time the set utility improves the best found so far. This guarantees that a possible overestimation of the costs is not preventing any promising set to be checked.

A part of this work entails changing relaxed-plan distances to satisficing-plan distances. The new estimations are obtained by using a width-based planner (Lipovetzky and Geffner 2012). The *width* of a planning problem P, $w(P)$ is a measure of the complexity of that problem. If $w(P) = i$, P can be solved optimally in exponential time in i. It turns out, that independently of the problem, most benchmark domains exhibit a low w when goals are a single fluent, which makes them solvable in low polynomial time by algorithms that prune nodes considering their *novelty*. If any fluent is true in a state for the first time, the state's novelty is 1. If no single fluent is novel but a pair of them appear together for the first time, the novelty is 2 and so on (the higher the novelty the less novel the state is). Novelty can be used to treat nodes as duplicated if their novelty is higher than a given value, as performed by the family of width-based planners (Frances et al. 2018; Lipovetzky and Geffner 2012, 2017b,a). Some of these planners, though incomplete, are polynomial, which makes them good candidates to be used to estimate the costs of achieving the goals in OSP.

On-Demand and One-Goal-Plan Distances

RPLD computes all distances, regardless of whether they are actually needed in the search or not. While Δ_{Ix} needs to be calculated for all goals, Δ_{xy} is needed only if x is added to the set, and just for not already selected goals y. This strategy, we call Relaxed Plan On-demand Linear Distances (RPOLD), significantly reduces the number of computed distances, allocating more time for the next steps of the algorithm. Meanwhile, NPLD (Novelty-based Polynomial Linear Distances) computes the distances on-demand too, but using satisficing plans found by a novelty-based planner.

Using satisficing plans for Δ_{Ix} produces values that are always equal to or higher than the optimal costs and are expected to be higher than the ones derived from relaxed plans. This mitigates one of the drawbacks of RPLD, as in numerous cases the selected set of goals remains oversubscribed.

Another advantage of working with satisficing plans is that the plan to calculate Δ_{Ix} is a valid one, with utility $u(x)$, so if its cost is lower than c_b it is a first solution. The same applies, when selecting the second goal. The concatenation of the plans found to calculate Δ_{Ix} and Δ_{xy} is a plan achieving the second goal and, quite often, also the first one.

Using satisfacing plans also poses some drawbacks. First, the time required to find them is notably higher compared to computing relaxed plans, even when utilizing polynomial planners. Additionally, these polynomial planners are incomplete, resulting sometimes in the distance being erroneously set to infinity. Furthermore, it is important to note that finding a plan that achieves x and subsequently pursuing y may result in no viable plan for both goals together.

RPLD is asymptotically complete and optimal if the external planner is complete (García-Olaya, de la Rosa, and Borrajo 2021) and so is RPOLD. NPLD adds a source of incom-

pleteness, as the polynomial planners used to calculate the distances are incomplete. But like RPLD, it will eventually try all the sets of goals, despite the values of the distances, so it is also asymptotically complete and optimal.

RPLD's complexity is quadratic in the number of relaxed plans. In the worst case, when all the goals are selected, NPLD needs to calculate n plans for Δ_{Ix}, $n-1$ for the first selected goal, $n-2$ for the second, etc. So it is also $O(n^2)$, but in the number of satisficing plans, which, if using a polynomial planner to calculate the distances, is also polynomial.

Experimental Results

As in (García-Olaya, de la Rosa, and Borrajo 2021), the performance of the three approaches is evaluated in the 14 domains of the IPC 2011. To ensure fairness, the SCANALYZER domain has been excluded due to RPLD's reported parser issues. High, medium, and low oversubscription versions of each original problem, with cost bounds of 25%, 50%, and 75% of its best-known cost, are used. Additionally, two distinct goal utility schemata are considered: `util1`, where all goals are assigned unit utility, and `util10`, where each goal is randomly assigned a utility value between 1 and 10.

To ensure the differences between the three approaches are due only to the values of the distances, all of them use the same external planner: BFWS-polynomial (Frances et al. 2018). It is incomplete, but yields better results than DUAL-BFWS, originally used by RPLD. As in the RPLD paper it is given 90 seconds to find a plan. BFWS-polynomial is also the planner used for NPLD distances calculation.

The experiments have been performed on an Intel Xeon CPU X3470 at 2.93 GHz with Ubuntu 20.04.6, 30 GB of RAM memory and 1800 seconds of maximum time[1].

Distance Calculation and Goal Selection

RPLD fails calculating distances in 96 problems, all of them in VISITALL. RPOLD lazy distance calculation reduces that number to 88. Meanwhile, NPLD cannot calculate distances in 99 problems, most of them in VISITALL, but also in OPENSTACKS, TRANSPORT and ELEVATORS. Despite being much slower calculating distances, NPLD is able to select goals in VISITALL problems where RPOLD is not. In that domain RPOLD highly underestimates the real cost of achieving the goals, adding many more goals than NPLD and exhausting the time while doing it. This behavior is also present in ELEVATORS.

Our initial hypothesis is that NPLD will generate sets with lower oversubscription than RPLD/RPOLD. Notice that both RPLD and RPOLD will always select the same sets of goals, but RPOLD will be able to do it in more problems. To validate this hypothesis, we attempted to find a plan for the sets selected by both approaches allowing different planners (specifically, BFWS-polynomial, DUAL-BFWS and BFWS(f_5)-landmarks) 1800 seconds once the goals are selected. Table 1 presents the percentage of problems for which none of the planners managed to find a plan. While this outcome does not definitively confirm that the set is oversubscribed, it serves as a reasonable proxy. Allocating more time or using alternative planners could potentially yield a solution.

[1]The code and the detailed results can be found at https://github.com/agolaya/ICAPS25_OSP

	util1		util10	
	NPLD	RPOLD	NPLD	RPOLD
barman	40%	50%	17%	30%
elevators	62%	92%	13%	65%
floortile	92%	88%	37%	18%
nomystery	0%	30%	0%	20%
openstacks	2%	0%	2%	7%
parcprinter	42%	53%	28%	33%
parking	40%	63%	3%	8%
pegsol	87%	83%	27%	28%
sokoban	28%	43%	20%	53%
tidybot	13%	67%	12%	58%
transport	33%	100%	13%	93%
visitall	70%	100%	52%	42%
woodworking	0%	0%	2%	0%
average	40%	57%	18%	35%

Table 1: Percentage of (likely) still oversubscribed problems.

The results confirm our hypothesis, when all goals have the same utility, NPLD returns an average of 40% still oversubscribed problems (57% in the case of RPOLD). RPOLD is less oversubscribed in some domains, but by a small difference. When the goals have different utility, the sets tend to be less oversubscribed in both cases. The reason is that goals are selected by utility, using the distances just in case of ties, which usually means fewer goals are selected. In this case NPLD is also less oversubscribed than RPOLD.

Times required for calculating distances have not been compared, as NPLD necessitates invoking an external planner and writing a new problem to disk, whereas RPOLD executes the entire process internally. Nevertheless, obtaining a satisficing plan is typically slower than generating a relaxed one. Consequently, RPOLD is expected to be faster. Also, distance calculation time is not as relevant as time needed to find the first solution, which is presented in the next section.

Utility of the Plans

To compare the utility of the plans we use the IPC score: For each problem, a planner is assigned a score between 0 and 1, resulting in dividing its utility by the maximum utility found. As there are 13 domains and 20 problems per domain, the maximum accumulated score is $20 \cdot 13 = 260$.

Table 2 compares the scores of the first solution found by the planners, stopping them once a solution is found: the planner will calculate the distances, use them to select goals, plan for them for 90 seconds and stop if a plan has been found. If no plan is found, the last added goal will be removed and planning will be performed again until returning a solution or discarding all the goals.

NPLD yields better results than RPLD/RPOLD in all the settings. The difference is bigger in 25% cost problems, where fewer goals can be selected and thus fewer distances need to be calculated. RPOLD only slightly improves the results

	util1			util10		
	25%	**50%**	**75%**	**25%**	**50%**	**75%**
RPLD	203.9	203.7	206.8	189.1	201.9	209.8
RPOLD	203.9	203.7	206.8	191.0	203.8	211.6
NPLD	**220.1**	**216.1**	**211.6**	**218.2**	**216.4**	**215.8**

Table 2: First plan scores. Best scores highlighted in bold.

	util1			util10		
	25%	**50%**	**75%**	**25%**	**50%**	**75%**
barman	0.7	-0.5	-0.1	-0.1	0.4	-0.2
elevators	-2.3	-2.5	-3.6	0.6	-2.0	-3.6
floortile	-0.4	0.5	0.9	-0.4	0.9	0.3
nomystery	0.3	-1.9	-1.9	2.6	-3.0	-3.1
openstacks	6.8	6.2	3.6	9.8	5.7	5.1
parcprinter	0.1	0.2	-0.1	0.0	1.3	0.5
parking	0.5	0.3	0.3	0.0	-0.2	-0.6
pegsol	0.3	0.3	-0.2	0.0	0.5	0.3
sokoban	0.4	-0.7	-1.7	6.4	-1.1	-2.4
tidybot	1.1	3.2	0.7	-0.3	2.0	1.1
transport	-4.0	-4.0	-5.5	-2.0	-3.9	-4.9
visitall	14.0	14.9	15.1	11.9	11.5	11.7
woodworking	-2.2	-3.4	-4.0	0.3	0.4	0.1

Table 3: Difference in score for the first plan between NPLD and RPOLD. A positive value means NPLD is better.

of RPLD in VISITALL with `util10`, where on-demand distances allows it to solve some more problems.

To assess whether the better score of NPLD spans across all the domains or is due to it vastly outperforming RPOLD in a few of them, a detailed description is presented in Table 3. NPLD yields a higher utility in 54% of the settings, RPOLD in 42%, with a tie occurring in the remaining 4%.

Figure 1 illustrates the percentage of problems solved within a given time frame. It is evident that NPLD not only achieves superior results compared to RPOLD but also requires less time overall. This is surprising as RPOLD, using relaxed plans calculated internally, computes distances faster than NPLD. But the time saved in distances calculation is quite often spent in refining the more oversubscribed sets, thus confirming our hypothesis that it is worth spending more time in the cost estimations. This not only produces plans of better quality but also does it slightly faster.

Table 4 shows the scores when planners are given 1800 seconds, thus looking for new goal sets after the first plan has been found. Here the three approaches behave more similarly, and actually both RPLD and RPOLD get better score for 75% oversubscription, due mainly to NPLD exhausting time calculating distances. RPOLD obtains higher utility in 59% of the configurations, NPLD in 29% and there is a tie in the remaining 12%. Actually if VISITALL is excluded, RPOLD's score is higher in all the settings. Even if RPOLD's first solutions take slightly longer to find, it is able to calculate remaining distances faster and to explore more goal combinations. The main differences appear in OPENSTACKS, where RPOLD significantly overestimates distances, which results in low quality first plans. But it leverages the extra time to explore sets that initially appeared oversubscribed, only to find they are not. Meanwhile, NPLD struggles calculating

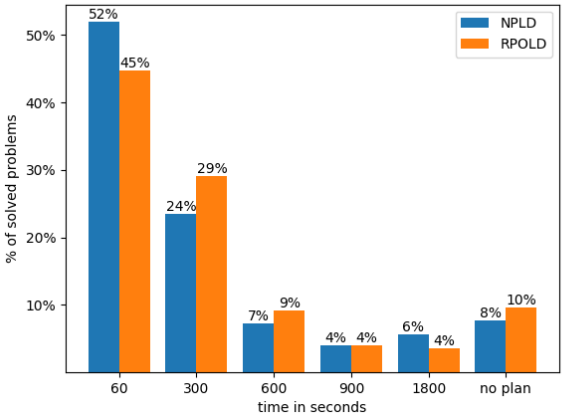

Figure 1: Percentage of problems solved in less than a time.

	util1			util10		
	25%	**50%**	**75%**	**25%**	**50%**	**75%**
RPLD	232.1	236.3	235.7	229.4	237.9	236.6
RPOLD	232.4	236.5	**236.5**	231.3	240.1	**239.5**
NPLD	**238.3**	**238.6**	233.8	**241.1**	**241.9**	238.2

Table 4: Summary of the final score.

distances in this domain. As expected, NPLD is better with more oversubscription (25%), where fewer goals can be selected. In the domains where it struggles selecting goals this allows it to solve one or two more problems.

Conclusions and Future Work

In this paper we have evaluated the improvements arising from calculating on-demand distances and using satisficing plans to estimate the costs of achieving goals in oversubscription planning. Those distances are used to select an ideally non-oversubscribed set of goals that can be planned by any classical planner able to handle the cost bound.

Results show that satisficing plans yield fewer oversubscribed sets of goals than using relaxed plans, thus finding first plans of higher utility slightly faster. When extra time is allotted to iteratively find sets of goals that improve the utility of the first plan, the differences between using satisficing or relaxed plans for costs estimations are smaller, except for some domains like VISITALL. Differences in the first solution utility between lazy or exhaustive calculation of distances are also small except for some problems in some domains.

The next step will be to explore a hybrid approach that combines satisficing and relaxed plans. The idea is to calculate the distances using satisficing plans until a given level, and at some point, if it is unlikely the selection process will finish in time, switching to calculating them with relaxed plans.

Acknowledgments

This work was partially funded by grants PID2021-127647NB-C21 and PDC2022-133597-C43 from MICIU/AEI/10.13039/501100011033, by the ERDF "A way of making Europe" and Next Generation EU/ PRTR, by the call for the Universidad Carlos III de Madrid for Grants for the recertification of the Spanish university system for 2021-2023, based on Royal Decree 289/2021 and by the Madrid Government under the Multiannual Agreement with UC3M in the line of Excellence of University Professors (EPUC3M17) in the context of the V PRICIT (Regional Programme of Research and Technological Innovation)

References

Baier, J. A.; Bacchus, F.; and McIlraith, S. A. 2009. A Heuristic Search Approach to Planning with Temporally Extended Preferences. *Artificial Intelligence*, 173(5-6): 593–618.

Benton, J.; Coles, A.; and Coles, A. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2–10. São Paulo, Brazil.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2): 165–204.

Do, M. B.; Benton, J.; van den Briel, M.; and Kambhampati, S. 2007. Planning with Goal Utility Dependencies. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 1872–1878. Hyderabad, India.

Domshlak, C.; Hoffmann, J.; and Katz, M. 2015. Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence*, 221: 73–114.

Domshlak, C.; and Mirkis, V. 2015. Deterministic Oversubscription Planning as Heuristic Search : Abstractions and Reformulations. *Journal of Artificial Intelligence Research*, 52: 97–169.

Frances, G.; Geffner, H.; Lipovetzky, N.; and Ramírez, M. 2018. Best-First Width Search in the IPC 2018: Complete, Simulated, and Polynomial Variants (planner abstract). In *IPC 2018 planner abstracts*, 22–26.

García-Olaya, A.; de la Rosa, T.; and Borrajo, D. 2021. Selecting goals in oversubscription planning using relaxed plans. *Artificial Intelligence*, 291: 103414.

Katz, M.; and Keyder, E. 2022. A* Search and Bound-Sensitive Heuristics for Oversubscription Planning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 9813–9820. Virtual.

Katz, M.; Keyder, E.; Winterer, D.; and Pommerening, F. 2019. Oversubscription Planning as Classical Planning with Multiple Cost Functions. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 237–245. Berkeley, USA.

Lipovetzky, N.; and Geffner, H. 2012. Width and serialization of classical planning problems. In *Frontiers in Artificial Intelligence and Applications*, volume 242, 540–545. IOS Press.

Lipovetzky, N.; and Geffner, H. 2017a. A Polynomial Planning Algorithm that Beats LAMA and FF. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 195–199. Pittsburgh, USA.

Lipovetzky, N.; and Geffner, H. 2017b. Best-first width search: Exploration and exploitation in classical planning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 3590–3596. San Francisco, USA.

Muller, D.; and Karpas, E. 2018. Value Driven Landmarks for Oversubscription Planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 171–179. Delft, The Netherlands.

Refanidis, I.; and Vlahavas, I. 2001. The GRT planning system: Backward heuristic construction in forward state-space planning. *Journal of Artificial Intelligence Research*, 15: 115–161.

Sanchez-Nigenda, R.; and Kambhampati, S. 2005. Planning Graph Heuristics for Selecting Objectives in Oversubscription Planning Problems. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 192–201. Monterey, USA.

Smith, D. 2004. Choosing Objectives in Over-Subscription Planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 393–401. Whistler, Canada.

Speck, D.; and Katz, M. 2021. Symbolic Search for Oversubscription Planning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 11972–11980. Virtual.

van den Briel, M.; Sanchez, R.; Do, M. B.; and Kambhampati, S. 2004. Effective Approaches for Partial Satisfaction (Over-Subscription) Planning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 562–569. San Jose, California.