Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)

Optimising the Stability in Plan Repair via Compilation

Alessandro Saetti, Enrico Scala*

Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Italy
alessandro.saetti@unibs.it, enrico.scala@unibs.it

Abstract

Plan repair is the problem of solving a given planning problem by using a solution plan of a similar problem. This paper presents the first approach where the repair has to be done optimally, i.e., we aim at finding a minimum distance plan from the input plan; we do so by introducing a simple compilation scheme that converts a classical planning problem into another where optimal plans correspond to plans with the minimum distance from an input plan. Our experiments using a number of planners show that such a simple approach can solve many problems optimally and more effectively than replanning from scratch for a large number of cases. Also, the approach proves competitive with LPG-adapt.

Introduction

Agents acting in real-worlds have to deal with uncertainty, therefore any plan can become invalid at some point. An approach to address such a problem is to either anticipate all possible contingencies at planning time (e.g., through conformant or contingent planning models (Smith and Weld 1998; Bonet 2010)), or deal with them as soon as something disruptive actually arises. When, however, there is no model about the uncertainty, or the number of unexpected situations is not bounded, it is nonetheless necessary to have some mechanism to come up with a new course of actions. A solution to such a problem is replanning. That is, as soon as the agent recognises that its plan does not work anymore, it formulates a new problem and plans from that point onward.

Although replanning can work well in some situation (Yoon, Fern, and Givan 2007; Ruml et al. 2011) and trying to reuse a plan is PSPACE-complete (Nebel and Koehler 1995), it is well known that trying to fix the current course of actions can be much more effective in practice (Gerevini and Serina 2010; Scala and Torasso 2014). Not only can the plan be more easily recoverable, but also the number of modifications to apply can be limited, therefore optimising the stability of the system. The plan stability is important when humans have already validated the planning activities under execution and the effort required for such a validation is considerable. In this case, stable plans reduce the cognitive load on human observers by ensuring coherence and consistency of behaviours (Fox et al. 2006). Previous solutions to such a problem have however no guarantees that the recovered plan is the most stable plan that can be computed (Scala and Torasso 2015; Fox et al. 2006; Gerevini and Serina 2010; Garrido, C., and Onaindia 2010; van der Krogt R. and de Weerdt M. 2005; Goldman, Kuter, and Freedman 2020; Höller et al. 2018; Scala 2014). Following on this line of research, in this paper we take the problem of stability as the primary objective of the plan repair problem. In particular we present a compilation-based approach that, given a planning problem and a plan, solves the plan repair problem with the guarantee to find plans that are at the minimum distance w.r.t. the input plan. We do so by making heavy use of a cost function that captures the implications of selecting actions that do no belong to the input plan, and that captures whenever the agent is not using actions in the input plan. Our compilation produces a classical planning problem that can be handled by any cost-sensitive planner.

To understand the practicability of our compilation, we report on an extensive experimental analysis comparing various optimal and satisficing planners with and without our compilation, over the planning problems from the 2018 edition of the planning competition, as well as with LPG-adapt, a state-of-the-art system that natively repairs plans.

Background

Let F be a set of facts; an action a is a pair $\langle \mathrm{pre}, \mathrm{eff} \rangle$ where pre is a formula over F indicating the precondition of a; eff is a set of positive and negative literals over F denoting the effects of a. With eff^+ and eff^- we indicate the positive and the negative effects of a. A state s is a set of facts from F with the meaning that if $f \in s$ then f is true in s, otherwise it is false. An action a is applicable in s if and only if $s \models \mathrm{pre}(a)$. The execution of an action a in s, denoted by $s[a]$, generates a new state s' such that $s' = (s \setminus \mathrm{eff}^-(a)) \cup \mathrm{eff}^+(a)$.

A classical planning problem $\mathcal{P}$ is the tuple $\langle F, A, I, G, c \rangle$, where F is a set of facts, A is a set of actions defined over F, $I \subseteq F$ is a state called the initial state, G is a formula over F, $c : A \mapsto \mathbb{R}_{\geq 0}$ is a function associating non-negative costs to actions. A plan π is a sequence of actions. The application of $\pi = \langle a_1, \ldots, a_n \rangle$ in a state s_0 gives the sequence of states $s_0[\pi] = \langle s_0, \ldots, s_n \rangle$

*Corresponding author. Email: enrico.scala@unibs.it

where $s_i = s_{i-1}[a_i]$ for all $1 \leq i \leq n$. Plan π is said to be a solution for $\mathcal{P}$ from $I = s_0$ if and only if the sequence of states $I[\pi] = \langle s_0, \ldots, s_n \rangle$ is such that for all $1 \leq i \leq n$ we have that $s_{i-1} \models \text{pre}(a_i)$ and $s_n \models G$. The cost of a plan π is $cost(\pi) = \sum_{a \in \pi} c(a)$. A plan π is said to be optimal if it is a solution and there is no plan π' such that π' is a solution for $\mathcal{P}$ and $cost(\pi) > cost(\pi')$.

In this work, we use a simple notion of plan distance, which considers a plan more stable if it is closer to the input plan in terms of number of different actions they contain, while we ignore the action ordering in plans. This is the same notion of plan distance proposed by Fox et al. (2006). Formally, let π and π' be two plans, the distance D between π and π' is defined as $D(\pi, \pi') = |\pi \setminus \pi'| + |\pi' \setminus \pi|$. Operator $\setminus$ is defined so that $\pi \setminus \pi'$ contains $m - l$ instances of action a iff π and π' respectively contain m and l instances of a and $m > l$; $\pi \setminus \pi'$ contains 0 instances of a otherwise.

A repair problem combines a planning problem and a plan attempting to solve such a problem. Formally, a repair planning problem is the pair $\langle \mathcal{P}, \pi \rangle$, where $\mathcal{P}$ is a planning problem and π is some sequence of actions from actions in $\mathcal{P}$. A plan solves the repair problem if and only if it solves $\mathcal{P}$, too. What differs from the planning problem is the notion of optimality. A plan is said to be optimal for a repair problem if it is the one that minimises the distance from the input plan π and solves $\mathcal{P}$. That is: $\pi^* = \arg\min_{\pi' \text{ solves } \mathcal{P}} D(\pi, \pi')$.

Solving the Optimal Repairing Problem

This section presents a compilation that takes in input a repair problem, and generates a novel classical problem whose optimal solutions are optimal solutions for the repair planning problem. Our compilation creates a number of copies for each action in the plan, and customises the cost function for keeping track of those actions undermining the optimality of the solution. The compilation also makes use of a number of additional, dummy predicates, whose purpose is to monitor the already executed actions. In what follows, we formally explain the compilation.

We use two functions to simplify notation. Function B counts the number of occurrences of input action a *Before* step i in the plan; formally, $B : A \times \mathbb{N} \to \mathbb{N}$. Function $M : A \to \mathbb{N}$ returns the number of repetitions of action a. We call our compilation RESA (REpair for StAbility).

Definition 1 (RESA Compilation)**.** *Let $\mathcal{P} = \langle F, A, I, G, c \rangle$ be a planning problem, and $\pi = \langle a_1, \ldots, a_n \rangle$ be a sequence of actions in A.* RESA *takes in input $\mathcal{P}$ and π, and generates a new planning problem $\mathcal{P}' = \langle F', A_0 \cup A_1 \cup A_2 \cup A_3 \cup \{switch\}, I', G', c' \rangle$ such that:*

$$F' = F \cup \{w\} \cup \bigcup_{i \in \{1,\ldots,n\}} d_i \cup \bigcup_{a \in \pi} \{p_a^i \mid 0 \leq i \leq M(a)\}$$

$$I' = I \cup \{w\} \cup \bigcup_{a \in \pi} p_a^0$$

$$A_0 = \bigcup_{a_i \in \pi} \langle \text{pre}(a) \wedge w \wedge p_a^{B(a,i)}, \text{eff}(a) \cup \{p_a^{B(a,i)+1}, \neg p_a^{B(a,i)}, d_i\} \rangle$$

$$A_1 = \bigcup_{a \in A \setminus set(\pi)} \langle \text{pre}(a) \wedge w, \text{eff}(a) \rangle$$

$$A_2 = \bigcup_{a \in set(\pi)} \langle \text{pre}(a) \wedge w \wedge p_a^{M(a)}, \text{eff}(a) \rangle$$

$$A_3 = \bigcup_{i \in \{1,\ldots,n\}} \langle \neg w \wedge \neg d_i, \{d_i\} \rangle$$

$$switch = \langle w, \{\neg w\} \rangle$$

$$G' = G \wedge \bigwedge_{i \in \{1,\ldots,n\}} d_i$$

$$c'(a) = \begin{cases} 0 & \text{if } a \in A_0 \cup \{switch\} \\ 1 & \text{if } a \in A_1 \cup A_2 \cup A_3 \end{cases}$$

Intuitively, the compilation reshapes the planning problem in such a way that all actions that do not contribute in increasing the distance from the previous plan are given cost 0. Action instances that instead were not in the plan (the actions set A_1) or that were in the plan but we have already used them (set A_2), or that were in the plan but are not going to be considered for the new plan (set A_3) are given cost 1. Indeed, the goal formula requires that, besides achieving the problem goals, all actions of the input plan are processed. This is achieved by formulating a fact d for each action within the starting plan. Such a fact can either be made true by actions from set A_0 whose cost is equal to 0 – indeed that corresponds to the case in which we did replicate what the plan was before, or actions from A_3 whose cost is equal to 1. Actions from A_3 are the *give-up* actions, that is, they emulate whether the planner gave up in trying to pick actions from the input plan and for that it pays an extra cost of 1 for each one of them. These dummy actions share some analogy with previous work to compile soft goals away (Keyder and Geffner 2009). In order to consider whether some action instance has been already considered we make use of the aforementioned functions B and M. These functions make it possible to create as many p_a predicates as needed to keep track of the number of instances of used action a, and to monitor whether limit $M(a)$ has been hit. This way, any new occurrence of some action in A_2 will increase the distance from the starting plan. The *switch* action is what finalises the search of the plan and starts the collection of the *give-up* actions from A_3. Indeed, none of the actions is executable after *switch* but those in A_3.

Properties of RESA

In this section we study some property of RESA. In particular we prove that RESA is sound, complete and always generates optimal solutions for the repair problem it is encoding. Moreover, RESA size is polynomial in the input task.

Theorem 1 (RESA is sound, complete and optimal)**.** *Let $\mathcal{R} = \langle \mathcal{P}, \pi \rangle$ be a plan repair problem.* RESA *transforms $\mathcal{R}$ into a problem $\mathcal{P}'$ that is solvable if and only if so is $\mathcal{R}$. Moreover, the optimal solution for $\mathcal{P}'$ equates to that of the solution for $\mathcal{R}$ that minimises the distance from π.*

Proof Sketch. (Soundness) Observe that the actions formulated by RESA do not alter the semantics of the original actions. Indeed all such actions require the precondition of the

Optimal setting	A* (Blind)		A* (h_{max})		Delfi1	
Domain	RS	RESA	RS	RESA	RS	RESA
AGRICOLA (48)	0	**8**	0	**10**	**39**	14
CALDERA (45)	18	**45**	18	**45**	39	**45**
DATA-NET (48)	16	**48**	27	**48**	39	**48**
NURIKABE (42)	30	**37**	30	**42**	**36**	18
SETTLERS (30)	19	**25**	22	**28**	25	**28**
SPIDER (51)	**17**	**17**	19	**41**	29	**41**
TERMES (54)	**18**	9	9	**20**	**36**	35
D-1 (106)	39	**74**	42	**85**	**82**	**82**
D-2 (106)	42	**66**	43	**77**	**82**	76
D-5 (106)	37	**49**	40	**72**	**79**	71
Total (318)	118	**189**	125	**234**	**243**	229

Satisficing setting	Lama		BFWS	
Domain	RS	RESA	RS	RESA
AGRICOLA (48)	**37**	16	**16**	10
CALDERA (27)	**27**	**27**	**27**	**27**
DATA-NET (57)	15	**55**	**28**	22
NURIKABE (57)	31	**56**	31	**57**
SETTLERS (57)	47	**54**	10	**15**
SPIDER (54)	**48**	23	**35**	33
TERMES (48)	**42**	32	**26**	11
D-1 (116)	84	**89**	56	**59**
D-2 (116)	83	**86**	56	**60**
D-5 (116)	80	**88**	**61**	56
Total (348)	247	**263**	173	**175**

Table 1: Coverage Analysis. Each entry of the table corresponds to the number of problems solved by the system identified by the column using RS or RESA. D-{1,2,5} is a regrouping of the instances that considers all instances computed by injecting 1, 2, or 5 random actions in sequence. The number of problems is in parenthesis. Bolds are for best performers

Optimal setting	A* (Blind)		A* (h_{max})		Delfi1	
Domain	RS	RESA	RS	RESA	RS	RESA
agricola	–	–	–	–	30.00	**5.67**
CALDERA	7.89	**0.00**	8.56	**0.00**	3.97	**0.00**
DATA-NET	6.13	**1.75**	7.56	**1.78**	8.85	**1.82**
NURIKABE	17.03	**2.31**	16.23	**2.40**	14.92	**3.67**
SETTLERS	9.94	**1.82**	11.50	**2.09**	8.88	**2.08**
SPIDER	64.33	**1.67**	35.95	**11.32**	36.14	**11.79**
TERMES	18.56	**0.56**	19.78	**0.67**	29.10	**0.65**

Satisficing setting	Lama		BFWS	
Domain	RS	RESA	RS	RESA
AGRICOLA	49.20	**8.67**	52.25	**3.75**
CALDERA	21.52	**3.26**	24.11	**0.78**
DATA-NET	107.20	**53.47**	85.64	**41.45**
NURIKABE	65.48	**13.48**	83.35	**22.87**
SETTLERS	110.04	**88.42**	83.29	**19.14**
SPIDER	324.39	**245.09**	270.75	**233.36**
TERMES	**604.69**	1160.22	**575.00**	589.89

Table 2: Plan Distance Analysis. Each entry of the table corresponds to an average of the plan distances across problems solved by both RS and RESA. Bolds are for best performers

original actions to hold before their execution, and the original effects are preserved. The remaining preconditions and effects only monitor whether the action instance is playing the role of an action that was in the input plan, or not.
(Completeness) Observe that for any valid plan π' of $\mathcal{R}$ there is one plan in the compiled one. Each action in π' has a copy either in set A_0 or in set A_1 or in set A_2 that is applicable.
(Optimality) Let π' be an optimal plan for $\mathcal{R}$. We can construct a solution for $\mathcal{P}'$ by choosing for each action in π' one in $A_0 \cup A_2$ according to the prefix of π' if the action is also in π, one in A_1 otherwise; subsequently, adding action *switch* and one action in A_3 for each action instance in π that is not in π'. It is easy to see that the resulting cost of the solution will reflect the overall cost of π'. □

Theorem 2. RESA *is linear on the size of* $\mathcal{R}$.

Proof sketch. Observe that there are only $O(\pi)$ new predicates and actions. □

Experimental Analysis

Our experimental analysis evaluates the performance of RESA comparing it with both replanning from scratch (hereinafter RS), and LPG-adapt, which is the state-of-the-art approach to plan repair; LPG-adapt adapts plans through a refinement approach in the plan spaces. It is designed to take plan stability into account (Fox et al. 2006).

In our comparison against RS, we study RESA with both optimal and satisficing planners. With optimal planners, we evaluate the coverage of doing optimal plan repair against just doing optimal planning. The goal of this comparison is evaluating how harder/simpler the problem gets when we aim at finding plans of minimum distance from the input plan. Moreover, we also collect the actual plan distance found by the planners to show how different the solutions found by RESA w.r.t. just ignoring the input plan are. We used the same metrics for the satisficing planners. Also for the satisficing setting we aim at understanding how harder/simpler the satisficing problem gets when the search is driven by minimizing the distance from an input plan, and the effectiveness of the satisficing planners in computing solutions closer to the input plan. For optimal planning, we used A* with a simple blind heuristic, A* with the h_{max} heuristic, and Delfi1 (Katz et al. 2018), the winner of the 2018 edition of the Int. Planning Competition (IPC-18). For the satisficing setting we use Lama (Richter and Westphal 2010) and BFWS (Lipovetzky and Geffner 2017). For both systems we run only the first cost-sensitive iteration. Finally, we compare coverage and plan distance of LPG-adapt w.r.t. optimal planning and the anytime version of Lama.

We take all the domains and problems from the optimal and satisficing track of the IPC-18. For each instance problem we generate three plan repair problems modifying the initial state by randomly executing 1, 2 and 5 actions in sequence. As an input plan, we used the shortest plan among those generated during the competition. Ties among shortest plans are broken randomly. This gives us a grand total of

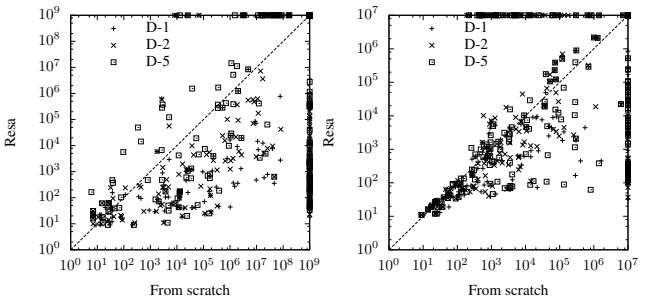

Figure 1: Scatter plotting the number of node expansions between RESA and replanning from scratch (RS). Each point is the pairwise comparison between RESA and RS with all optimal (left) and satisficing (right) planners. Full scale values are problems unsolved by one of the two systems.

666 instances. All experiments ran up to 1800 seconds with a memory cut of 8GB. Tests are performed on a single core of a 4 cores Intel(R) Xeon(R) 2.30 GHz.

RS vs RESA, Optimal Planning. In our comparison against RS, each experiment consists in launching each instance and each planner either with the original formulation of the problem or with that obtained by our compilation. Table 1 shows the coverage of A* ran with the blind and the h_{max} heuristic, and Delfi1 over both the original problem and the problem compiled by RESA. As it is possible to observe, RESA with A* performs much better than RS. To have a better understanding of the reasons why RESA performs so much better with A*, we also collected information on the number of expanded nodes during the search. This is reported in Figure 1. As it is possible to observe, RESA makes the search much simpler. Indeed, the planner is encouraged to explore all 0-cost path first, and, especially for small discrepancies, there is quite a gain of coverage. For Delfi1 we instead have mixed results. In AGRICOLA and NURIKABE, in particular, RS proves to be more beneficial than repairing. By looking at our raw data we observed that Delfi1 spends a lot of time in preprocessing. RESA generates ground problems, and this seems particularly problematic for the preprocessing step of Delfi1, to the point that with RESA even a much simpler search such as A* combined with h_{max} performs better than Delfi1. Our conjecture about the reason for the poorer performance of RESA even with Delfi1 seems supported by Figure 1. Indeed, in terms of number of expanded nodes, RESA with Delfi1 expands fewer nodes than RS, since the vast majority of the points are under the diagonal. As expected, the performance of RESA gets worse as the number of differences w.r.t. the original problems increases. This also indicates that the optimal repair problem becomes progressively more difficult as the number of such differences increases. Finally, the results in Table 2 show that in terms of plan distance RESA using optimal planners is always much better than replanning.

RS vs RESA, Satisficing Planning. Table 1 reports on the coverage of Lama and BFWS with or without the RESA compilation. Note that the compilation improves the coverage of Lama over the majority of domains, but AGRICOLA, SPIDER and TERMES. The problems of these domains have solution plans definitely longer than other domains. Our conjecture is that longer plans make the compilation more difficult, nullifying the guidance that such plans can provide in solving the problem. The results of RESA with BFWS are similar. Regarding BFWS and Lama, we have the similar problem we got with Delfi1. As Delfi1, Lama and BFWS are based on the Fast-Downward planning system (Helmert 2006), and such a system seems to be not well optimised to work over fully grounded problems. Interestingly, differently from the optimal setting, the satisficing repair problem does not become more difficult as the number of differences w.r.t. the original problem increases. Table 2 shows that for all domains but TERMES the plans computed by RESA are less distant from the input plan than RS. In TERMES, very long input plans give no fruitful information to the search.

	Solved		Distance		Solved		Distance	
Domain	LPG	RD	LPG	RD	LPG	RL	LPG	RL
AGRICOLA	**16**	14	126.5	**14.50**	16	**43**	136.1	**106.7**
CALDERA	–	**45**	–	–	–	**45**	–	–
DATA-NET	**48**	**48**	4.94	**1.85**	**48**	**48**	4.94	**2.73**
NURIKABE	–	**18**	–	–	–	**42**	–	–
SETTLERS	–	**28**	–	–	–	**28**	–	–
SPIDER	–	**41**	–	–	–	**31**	–	–
TERMES	**54**	35	0.80	**0.63**	**54**	48	**0.79**	108.0

Table 3: Coverage and plan distance between LPG-adapt and RESA using Delfi1 (RD) and Lama (RL) over the problems derived from the optimal track of IPC-18. Bolds are for best performers; "–" for unsupported.

LPG-Adapt vs RESA. We consider RESA with Delfi1 and Lama, the best performers among the used optimal and satisficing planners. We take the best solution found by LPG-adapt and RESA using Lama in anytime modality run up to 1800s. The results show that (Table 3), for all domains but TERMES, RESA with Delfi1 is competitive with LPG-adapt in terms of coverage, while guaranteeing the optimality of the solution for the repair problem. The plans computed by RESA with Delfi1 are up to one order of magnitude closer to the input plans (i.e., in AGRICOLA). RESA with Lama solves much more problems than LPG-adapt overall, but is not effective in TERMES. Missing results of Table 3 are due to LPG-adapt not supporting conditional effects. An extended analysis without conditional effects is needed to better compare LPG-adapt with RESA.

Conclusion

This paper faces the problem of optimal plan repair, which consists in finding a plan that solves the problem and also minimises the distance with respect to the input plan. We solved this problem through a compilation into classical planning. Our results show that not only is this approach possible, but also competitive with LPG-adapt, the state-of-the-art system for computing repair plans optimising stability. As a future work we want to investigate different metrics for stability and adaptations of RESA to such metrics.

Acknowledgements

We thank the anonymous reviewers for their insightful comments. The authors have been supported by AIPlan4EU, a project funded by EU Horizon 2020 research and innovation programme under GA n. 101016442 (since 2021)

References

Bonet, B. 2010. Conformant plans and beyond: Principles and complexity. *Artif. Intell.*, 174(3-4): 245–269.

Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan Stability: Replanning versus Plan Repair. In *ICAPS*, 212–221. AAAI.

Garrido, A.; C., G.; and Onaindia, E. 2010. Anytime Plan-Adaptation for Continuous Planning. In *Proc. of P&S Special Interest Group Workshop (PLANSIG-10)*.

Gerevini, A.; and Serina, I. 2010. Efficient Plan Adaptation through Replanning Windows and Heuristic Goals. *Fundam. Informaticae*, 102(3-4): 287–323.

Goldman, R. P.; Kuter, U.; and Freedman, R. G. 2020. Stable Plan Repair for State-Space HTN Planning. In *Proceedings of the ICAPS-20 Workshop on Hierarchical Planning (HPlan 2020)*, 27–35.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26: 191–246.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2018. HTN plan repair using unmodified planning systems. In *Proceedings of the 1st ICAPS Workshop on Hierarchical Planning (HPlan)*, 26–30.

Katz, M.; Sohrabi, S.; Samulowitz, H.; and Sievers, S. 2018. Delfi: Online planner selection for cost-optimal planning. *IPC-9 planner abstracts*, 57–64.

Keyder, E.; and Geffner, H. 2009. Soft Goals Can Be Compiled Away. *J. Artif. Intell. Res.*, 36: 547–556.

Lipovetzky, N.; and Geffner, H. 2017. Best-First Width Search: Exploration and Exploitation in Classical Planning. In *AAAI*, 3590–3596. AAAI Press.

Nebel, B.; and Koehler, J. 1995. Plan Reuse Versus Plan Generation: A Theoretical and Empirical Analysis. *Artificial Intelligence*, 76(1-2): 427–454.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *J. Artif. Intell. Res.*, 39: 127–177.

Ruml, W.; Do, M. B.; Zhou, R.; and Fromherz, M. P. J. 2011. On-line Planning and Scheduling: An Application to Controlling Modular Printers. *Journal of Artificial Intelligence Research (JAIR)*, 40: 415–468.

Scala, E. 2014. Plan Repair for Resource Constrained Tasks via Numeric Macro Actions. In *ICAPS*. AAAI.

Scala, E.; and Torasso, P. 2014. Proactive and Reactive Reconfiguration for the Robust Execution of Multi Modality Plans. In *ECAI*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, 783–788. IOS Press.

Scala, E.; and Torasso, P. 2015. Deordering and Numeric Macro Actions for Plan Repair. In *IJCAI*, 1673–1681. AAAI Press.

Smith, D. E.; and Weld, D. S. 1998. Conformant Graphplan. In *AAAI/IAAI*, 889–896. AAAI Press / The MIT Press.

van der Krogt R.; and de Weerdt M. 2005. Plan Repair as an Extension of Planning. In *Proc. of International Conference on Automated Planning and Scheduling (ICAPS-05)*, 161–170.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. FF-Replan: A Baseline for Probabilistic Planning. In *ICAPS*, 352. AAAI.