Plan Stability: Replanning versus Plan Repair

Maria Fox[1] and **Alfonso Gerevini**[2] and **Derek Long**[1] and **Ivan Serina**[2]
1: Department of Computer and Information Sciences, University of Strathclyde, Glasgow, UK
firstname.lastname@cis.strath.ac.uk
2: Department of Electronics for Automation, University of Brescia, Brescia, Italy
lastname@ing.unibs.it

Abstract

The ultimate objective in planning is to construct plans for execution. However, when a plan is executed in a real environment it can encounter differences between the expected and actual context of execution. These differences can manifest as divergences between the expected and observed states of the world, or as a change in the goals to be achieved by the plan. In both cases, the old plan must be replaced with a new one. In replacing the plan an important consideration is *plan stability*. We compare two alternative strategies for achieving the *stable* repair of a plan: one is simply to replan from scratch and the other is to adapt the existing plan to the new context.

We present arguments to support the claim that plan stability is a valuable property. We then propose an implementation, based on LPG, of a plan repair strategy that adapts a plan to its new context. We demonstrate empirically that our plan repair strategy achieves more stability than replanning and can produce repaired plans more efficiently than replanning.

1 Introduction

Plans are constructed using abstract models of the worlds in which they are to be executed. Inadequacy of the predictive model can mean that execution of the plan leaves the executive in a state that diverges from the expectations of the planner. The behaviour of other agents in the world can change the state in ways that the planner could not predict and the context in which the original plan was formed can change so that new goals must be satisfied.

Markov decision processes (Boutilier, Dean, & Hanks 1999), contingency plans (Pryor & Collins 1996) and Just-In-Case (Drummond, Bresina, & Swanson 1994) schedules are constructed to be inherently robust to uncertainty during execution of plans and schedules. However, none of these approaches have been applied in situations in which the goals themselves change. In principle, these approaches could be adapted to treat the goals as part of the environment to which they respond, but the size of the space of possible goals makes scaling problematic. We do not consider these approaches further in this paper.

A situation in which planning problems change during execution is in satellite observation scheduling (Frank, Gross, & Kürklü 2004), where new observation requests can arise during the execution of a plan. In such situations, the original plan cannot or should not be executed but should be adapted to respond to the new situation.

Many authors have considered the relative benefits of replanning and plan repair from the point of view of both theoretical (Nebel & Koehler 1995) and empirical efficiency (Gerevini & Serina 2000). We identify a new metric, *plan stability*, which we claim is important in situations where the safety of the executive, or of equipment deployed by the executive, is a major consideration in evaluating a plan. In such situations a repaired plan should be as close as possible to the original plan whilst achieving the specified goals (including new goals) from the current state. The extent to which this is achieved is a measure of the stability of the new plan. The concept of *minimal perturbation planning* is closely related to this idea (Kambhampati 1990; Simmons 1988).

In this paper, we contrast *plan repair* and *replanning* in terms of plan stability. By *plan repair* we mean the work of adapting an existing plan to a new context whilst perturbing the original plan as little as possible. By contrast, *replanning* is the work of generating a new plan from scratch without considering stability. We show that, when the execution context is not highly dynamic, the original plan can be a very good guide to the construction of a new plan. Of course, in highly dynamic situations stability might be impossible to achieve. However, we argue that where it can be achieved stability is an important property that allows confidence to be maintained in the safe operation of an executive within its environment.

The approach we consider builds on that presented in (Gerevini & Serina 2000) for repairing "quasi-solutions". As in that earlier work, the approach exploits the local search strategy of LPG (Gerevini, Saetti, & Serina 2003), by seeding the search with the existing plan and by using a modified evaluation function to guide the search for a new plan. Our current work makes three new contributions as follows:

- we explicitly model plan stability and use it as a metric in determining the benefits of the adaptive approach;
- we use much improved search techniques and a more powerful graph-based plan representation and

· we handle PDDL2.1 problems rather than STRIPS problems as in (Gerevini & Serina 2000).

We show, empirically, that this repair approach can lead to high quality plans in a shorter time, while preserving a higher degree of stability, than can be achieved by replanning. The speed of plan repair can be extremely important. If goals change at a time δ before the existing plan requires an action to be executed, then the construction of a new plan must complete within δ if it is to continue to make use of this action at this time. If the original plan was the product of a complex piece of planning reasoning then the replanning strategy could be faced with an expensive task that cannot complete in the δ time limit and will lose the opportunity to find a good solution. A significant feature offered by plan repair is that it can benefit from higher quality initial plans to arrive at higher quality repaired plans. The basic adaptation strategy is as presented in (Gerevini & Serina 2000), but improved in the ways described above.

We first define the problem we are interested in solving. We then define plan stability and explain why we consider it an important property. We proceed to describe the technical details of our approach to replanning, which builds on the core of LPG and to present a detailed evaluation with respect to the planning time, plan quality and plan stability in comparison with replanning. We show that, in the context of plan execution in dynamic situations, our approach to plan repair can offer significant advantages over replanning.

2 Dynamic Planning Problems

We are interested in the situation in which an initial plan has been constructed and the context in which it is being executed has deviated from the expected context. The deviation might be a discrepancy between the expected value of the state and its observed value due to uncontrollable features of the domain, or else it might be that the goals of our original plan have changed and we must meet additional new goals and, possibly, abandon other goals. Although an unexpected change in the context can arise at any point during the execution of a plan, the plan that has already been executed cannot be retracted, so we can always consider the problem as though the current state were the initial state and the remainder of the unexecuted plan were the whole of the original plan.

Thus, we consider the following situations: we begin with an initial state, a goal and a plan that achieves this goal from this state and then we are presented with a changed initial state, or a changed goal, or both. More formally:

Definition 1 *A* dynamic planning problem *is a tuple* (I, G, π_0, I', G') *where* π_0 *is a plan that achieves the goal* G *from the initial state* I *and* I' *is a new initial state from which the new goal state,* G'*, must be achieved.*

We are assuming that the domains we are working with are dynamic, but not volatile: changes may occur in the initial state or in the goals, but are relatively small in each increment of change. We do not expect that the initial state should change by more than a few literals or the goals change by more than a few conditions. Thus, we intend I and I' to be very similar and G and G' to be very similar. The same assumptions were made in (Gerevini & Serina 2000).

This problem is motivated by similar observations to those made by other authors, such as (Traum & Allen 1994; Horty & Pollack 2001; van der Krogt & de Weerdt 2005), who have proposed various strategies for managing it. It should be noted, however, that it is observed in (Cushing & Kambhampati 2005) that there are problems in which a plan failure cannot be adequately modelled by changes in initial and goal states, but actually by modified versions of operators in the planning domain.

3 Plan Stability

We use the term "plan stability" to refer to a measure of the difference a process induces between an original (source) plan and a new (target) plan. In general, we will be considering cases where the new plan is intended to solve a different, although related, problem to the one solved by the original plan. This means that there will inevitably be a difference between the plans. Measure the difference between two plans is hard: if one plan shares the same actions as another, but they are organised into a different ordering then it is not entirely obvious whether the plans should be considered similar or not. The ordering might be irrelevant, if it does not change the outcome of the plan, or else it might be very significant, if the change completely alters the effects of the plan.

We are interested in achieving plan stability in order to preserve the content of the information a plan contains. For this purpose, a plan should be considered more stable if it is close to another not only in terms of the steps it contains, but also in terms of the order of those steps. However, we wish to compare our technique to replanning. In order to avoid stacking the odds against replanning, we adopt a more relaxed measure of stability, which is intended to deemphasise the internal structure of the original plan and focus on the actions it contains.

We define a measure of the difference between two plans.

Definition 2 *Given an original plan,* π_0*, and a new plan,* π_1*, the difference between* π_0 *and* π_1*,* $D(\pi_0, \pi_1)$*, is the number of actions that appear in* π_1 *and not in* π_0 *plus the number of actions that appear in* π_0 *and not in* π_1*.*

The symmetry of this definition is important, as the following example shows. Suppose that an asymmetric definition is used, in which only the number of actions that appear in the new plan and not in the original plan are counted. Now, suppose a plan π_0 contains a sequence of 12 drive actions moving some package, `p`, from Rome to Glasgow. The actions are `(drive p rome florence)`, `(drive p florence bologna)`, `(drive p bologna brescia)`, ...`(drive p london glasgow)`. Suppose that we change the goal so that the package should be delivered to Edinburgh instead of to Glasgow. In the plan, π_1, that results from our adaptation strategy, the final drive action is replaced by `(drive p london edinburgh)`, whilst the rest of the plan remains unchanged. The value $D(\pi_0, \pi_1)$ is 1. In the plan, π_2, that results from replanning, the whole sequence of drive actions is replaced by `(fly p rome`

edinburgh). The value $D(\pi_0, \pi_2)$ is 1. It can be seen that under the asymmetric definition $D(\pi_0, \pi_1) = D(\pi_0, \pi_2)$ but plan π_2 is clearly substantially different from plan π_1. Although they have the same D value, π_2 cannot be said to be as stable with respect to π_0 as is π_1, since π_2 contains no actions in common with π_0 whilst π_1 contains 11 common actions. Under the symmetric definition $D(\pi_0, \pi_1) = 1$ whilst $D(\pi_0, \pi_2) = 12$, which is a much more realistic reflection of their relative stability.

It can be noted that traditional notions of plan quality must sometimes be traded off for plan stability. As observed in (Cushing & Kambhampati 2005), the quality of a plan that revises one communicated to other agents cannot be evaluated independently of prior *commitments*. If other agents have altered their plans to coincide with a communicated plan, revised plans that stable with respect to the communicated plan might be considered to be of higher quality than plans that are completely new even if, considered independently, the new plans are better.

We are now in a position to define plan stability:

Definition 3 *Given a dynamic planning problem (I, G, π_0, I', G'), a planning strategy, P_1, achieves greater* plan stability *than a planning strategy, P_2, if the plans produced by P_1 and P_2, π_1 and π_2 respectively, satisfy $D(\pi_0, \pi_1) < D(\pi_0, \pi_2)$.*

We will show that our plan repair strategy achieves greater plan stability than replanning, across a wide range of dynamic planning problems.

Our definition of stability is still rather simple, and whilst reasonable for some domains might not be for others. In future work we consider that it would be useful to study more than one plan stability measure, perhaps forming a hierarchy of measures of which the one in this paper forms a base case.

3.1 The Value of Plan Stability

We have two informal arguments:

1. Preserving plan stability reduces the cognitive load on human observers of planned activity, by ensuring coherence and consistency of behaviours, even in the face of dynamic environments. One aspect of this argument is that, where plans are used to convey intended actions from one agent to another, higher plan stability makes the agents more predictable and interaction with them potentially easier.
2. More stable plans offer greater opportunity for graceful elision of activities and less stress on execution components. The translation of successive plan steps into actual execution can be simplified by their executive smoothing the transitions between actions. An example of this process for movement planning can be seen in (Stulp & Beetz 2005). Greater stability makes it possible to anticipate these elisions more reliably. If changes in the environment might lead to sudden and significant changes in the structure of the plan being executed then such smoothing becomes much harder.

We also have arguments that we test empirically:

1. It is possible to generate plans more quickly by seeking to preserve plan stability than by replanning.
2. Preserving stability by exploiting plan repair allows a planner to generate high quality plans by exploiting the reasoning that has already been performed in generating the original plan and by continuing to meet commitments made by the original plan.

Our experiments and results are described in section 5.

4 Plan Adaptation for Plan Repair: Solving Dynamic Planning Problems

Various approaches are possible for tackling dynamic planning problems including, at one extreme, ignoring the existing solution and treating the problem as a new planning problem, with its own initial state and goal. Indeed, it has been demonstrated that attempting to reuse an existing plan as the starting point for construction of a new plan can be as expensive as starting with nothing (Nebel & Koehler 1995). Nevertheless, many authors have observed that plan repair can offer advantages in practice (Gerevini & Serina 2000; Veloso 1994; Kambhampati & Hendler 1992; Hanks & Weld 1995; Koenig, Furcy, & Bauer 2002; van der Krogt & de Weerdt 2005). Strategies vary from attempting to reuse the structure of an existing plan by constructing bridges that link together fragments of the plan that fail in the face of new initial conditions (Hammond 1990; Kambhampati & Hendler 1992; Hanks & Weld 1995) to more dynamic plan modification approaches that use a series of plan modification operators to attempt to repair a plan (van der Krogt & de Weerdt 2005; Koenig, Furcy, & Bauer 2002).

The approach we consider in this work is based on the use of LPG (Gerevini, Saetti, & Serina 2003). LPG is a local-search-based planner, that modifies plan candidates incrementally in a search for a candidate that contains no flaws. This approach to planning sits ideally with the problem of plan repair, since the approach can be seen as consisting entirely of an incremental plan repair strategy. However, we are interested in an additional aspect of the plan repair process, which is the stability of the plans it produces. For this reason, we have modified the behaviour of LPG slightly, to emphasise the importance of maintaining plan stability. We describe the modification in the following section.

4.1 Heuristic Evaluation for Plan Adaptation

The behaviour of LPG is controlled by an evaluation function that is used to select between different candidates in a neighbourhood generated for local search. LPG searches a space of partial plans where, at each search step, the elements in the search neighbourhood of the current (partial) plan π are the alternative possible plans repairing a selected flaw in π.

The elements of the neighbourhood are evaluated according to an *action evaluation function* E (Gerevini, Saetti, & Serina 2003). This function is used to estimate the cost of either adding ($E(a)^i$) or of removing ($E(a)^r$) an action node a in the partial plan π being generated or adapted. In order to properly manage dynamic planning problems the function E has been extended to include an additional evalua-

tion term that has the purpose of penalising the insertion and removal of actions that increase the distance of the current partial plan π under adaptation from the input plan π_0. In general, E consists of four weighted terms, evaluating four aspects of the quality of the current plan that are affected by the addition or removal of a:

$$E(a)^i = \frac{\mu_E}{max_{ET}} \cdot Execution\_cost(a)^i + \\ + \frac{\mu_T}{max_{ET}} \cdot Temporal\_cost(a)^i + \frac{1}{max_S} \cdot Search\_cost(a)^i + \\ + \frac{\mu_\Delta}{max_\Delta} \cdot (D(\pi_0, \pi + \pi^i_{\mathsf{R}}) - D(\pi_0, \pi))$$

$$E(a)^r = \frac{\mu_E}{max_{ET}} \cdot Execution\_cost(a)^r + \\ + \frac{\mu_T}{max_{ET}} \cdot Temporal\_cost(a)^r + \frac{1}{max_S} \cdot Search\_cost(a)^r + \\ + \frac{\mu_\Delta}{max_\Delta} \cdot (D(\pi_0, \pi + \pi^r_{\mathsf{R}} - a) - D(\pi_0, \pi))$$

The first three terms of the two forms of E are unchanged from the standard behaviour of LPG. The first term of E estimates the increase of the plan execution cost, the second estimates either the end time of a (for E^i) or the earliest time when all preconditions that would become unsupported by removing a from π could be supported again, and the third estimates the increase of the number of the search steps needed to reach a solution graph (see (Gerevini, Saetti, & Serina 2003) for a detailed description). The fourth term, used only for dynamic planning problems, is the new term, estimating how the proposed plan modification will affect the distance from π_0.

Each cost term in $E(a)^i$ is computed using two functions, EvalAdd(a) and EvalDel(a), which are fully described in (Gerevini, Saetti, & Serina 2003). Briefly, EvalAdd(a) returns an estimate of *Temporal_cost(a)*^{i} together with a relaxed plan, π^i_{R}, that contains a minimal set of actions for achieving, in the context of the current plan π, (1) the unsupported preconditions of a and (2) the set of preconditions of other actions in the current partial plan that would become unsupported by adding a to it. The initial state for π^i_{R} is a state derived by executing the actions in π up to the point where a would be inserted. π^i_{R} represents the portion on the plan that is likely to be added into π in the subsequent planning process to deal with the flaws introduced by a. π^i_{R} is calculated to give an estimate of the expected distance between the finished plan we can expect to be constructed and the result of modifying π with action a.

Similarly, EvalDel(a) returns a relaxed plan π^r_{R} containing a minimal set of actions required to achieve the preconditions that would become unsupported if a were removed from π, together with an estimation of *Temporal_cost(a)*^{r}.

The π_{R} plans are computed by an algorithm, called RelaxedPlan, formally described in (Gerevini, Saetti, & Serina 2003), that we have slightly modified to penalize the selection of actions decreasing plan stability. Informally, when we choose an action to achieve a subgoal in the relaxed plan, in addition to the preference criteria already used by RelaxedPlan, we prefer those actions that do not increase the last term of E. RelaxedPlan constructs the relaxed plan π_{R} through a backward process using the heuristic function *Bestaction* in order to select the actions to insert in π_{R} (Gerevini, Saetti, & Serina 2003). In the adaptation context we extend *Bestaction* by introducing a penalty term for a candidate action b, $\Delta(b)$, that evaluates the insertion/removal of b into/from π_{R} considering the elements of π_0, π and the current π_{R} itself:

$$\Delta(b) = \begin{cases} 1 & \text{if } D(\pi_0, \pi + \pi_{\mathsf{R}} + b) - D(\pi_0, \pi + \pi_{\mathsf{R}}) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Thus, we penalize the evaluation of b for π_{R} if its addition/removal increases the distance of $\pi + \pi_{\mathsf{R}}$ from π_0.

During adaptation, the last term of the evaluation function E is a measure of the increase in plan distance caused by adding or removing a: the difference between $D(\pi_0, \pi + \pi^i_{\mathsf{R}})$ and $D(\pi_0, \pi)$, or $D(\pi_0, \pi + \pi^r_{\mathsf{R}} - a)$ and $D(\pi_0, \pi)$, where $\pi + \pi^i_{\mathsf{R}}$ contains the new action.

The coefficients of these terms are used to normalize them and to weight them. Thus μ_E, μ_T and μ_Δ are non-negative coefficients that weight the relative importance of the execution, temporal and "differences" costs, respectively. Their values can be set by the user or automatically derived from the expression defining the plan metric for the problem. The factors $1/max_{ET}$, $1/max_S$ and $1/max_\Delta$ are used to normalize the terms of E to a value that is (upper) bounded by 1. The value of max_{ET} is defined as $\mu_E \cdot max_E + \mu_T \cdot max_T$ and max_E, max_T and max_Δ are respectively the maximum value of the first, second and fourth term of E over all elements of the neighborhood, multiplied by the number κ of flaws in the current partial plan. The term max_S is defined as the maximum value of $Search\_cost$ over all possible action insertions or removals that eliminate the flaw under consideration.[1] Without this normalization the first two terms of E could be much higher than the others, guiding the search towards good quality plans without paying sufficient attention to their validity and similarity with respect to π_0. However, especially when the current partial plan contains many flaws, we would prefer the search to give more importance to reducing the search cost, rather than on optimizing the quality of the plan.

LPG has an anytime behaviour. It can produce a succession of valid plans, where each successive plan improves upon its predecessor. The first plan generated is used to initialize a new search for a second plan of better quality, and so on. This is a process that incrementally improves the quality of the plans, and that can be stopped at any time to give the best plan computed so far (π_{best}). Each time we start a new search, π_{best} is used to initialize the data structures and some flaws are introduced in it by removing some actions. Similarly, during search, some random flaws are forced in the plan currently undergoing adaptation, when a valid plan that does not improve π_{best} is reached. This is a mechanism for leaving local optima.

To allow LPG to repair an input plan, an initial structure is constructed to represent the input plan. This requires a simple analysis of the input plan, to identify the causal structure

[1] The role of κ is to decrease the importance of the first, second and fourth optimisation terms when the current plan contains many flaws, and to increase it when the search approaches a valid plan.

in the plan. There can be choices to be made in this construction but we do not explore alternatives: in practice, it does not significantly affect behaviour which choice is made.

5 Empirical Evaluation

LPG is a stochastic planner: it exploits a local search behaviour that uses random restarts in a similar way to Walksat. As a consequence, the behaviour of LPG is different for different random number seeds. In order to carry out our evaluations we have therefore carried out 5 runs for each problem instance, with different seeds, and we take the median of these values for our plots below. The tests were performed on an AMD Athlon(tm) XP 2600+ (with an effective 2100 MHz rating) and 512 Mb of RAM. In these tests the original input plan was obtained by using TLPLAN (Bacchus & Kabanza 2000) unless otherwise specified (this allowed us to use a high quality input plan with comparatively low investment of initial computation time). Using a plan from a different planner also ensures that we are not artificially enhancing stability by relying on the way in which the planner explores its search space. It might appear that in contexts where stability is important, using the same planner to produce a plan is an obvious technique that might help. We show that, for LPG at least, this is not an effective strategy. Of course, deterministic planning algorithms might offer greater stability, but we believe that where planners are not influenced by existing plans, there is unlikely to be reliable stability in the behaviour.

5.1 Experimental Set-up

Our tests are conducted on a series of variants of problems from different domains, including DriverLog Time and STRIPS versions, Logistics Simple Time (both a typed and an untyped variant), ZenoTravel Time and Rovers Time. We have concentrated mainly on temporal variants, since the contexts in which we anticipate exploiting the plan repair strategy involve temporal domains. The tests are generally performed by taking a single problem from the benchmark test suite in each case and then methodically generating a series of variants for that problem. In each case, where we use this approach, we have taken a large problem instance as the base problem, in order to make the question of repair versus replanning an interesting one (for small problems, the difference between these strategies is negligible). The variant problems are generated by modifying the initial state and goals for the original problem. The modifications are performed randomly, although the number of modifications is increased systematically: we consider successively, zero through to five modifications to the goal set and, for each of these cases, successively one modification through to five modifications of the initial state. We create five variants for each combination of modifications, so that, for example, there are five different variants with one goal and one initial state modification. We also use the original problem as the 176th instance, to be used as a special case. Doing this shows the cost of initialising the data structures for adaptation, since the repair strategy does not have to perform any modifications for this problem.

Although we use only one base problem in each case, we have generated a large number of variants and we have considered problems from several different domains, so these results can be considered representative of the behaviour of the system for other similarly sized base problems. However, to confirm that the results are not an artifact of the particular problem instances chosen, we have adopted a different problem generation strategy for generating problem instances in the ZenoTravel and Rovers tests. Here we selected problems randomly from the benchmark suites, distributed across the smaller and larger problem instances, and generated a variant problem for each case. We used the same scheme as above to determine the combination of modification values for the initial state and goals, but selected the base problem to apply the modifications randomly.

Our first experiments demonstrate that plan repair, using our approach, generates plans faster than replanning. It is possible that the fast plan repair comes at a price in plan quality, as a plan is adapted to fit a new problem rather than constructed to fit the new problem directly. We therefore compare the best relative qualities of plans generated by repair and by replanning within a certain CPU-time limit. We then check that the plan repair process really maintains higher stability in the plans, since it might be that the plan seed provides a good starting point for the search for a new plan, but is completely replaced in the process of constructing the new repaired plan. Thus, we compare the distances between the final plans and the input plans for replanning and plan repair (where replanning is, of course, completely uninfluenced by the input plan).

5.2 Results

Our first results, shown in figures 1, 2, 3, 4, 5 and 6, examine plan repair versus replanning for variants of problems in six different sets of variants. In each case we show the time taken to produce a solution (this corresponds to the first plan generated by LPG). We then show, giving time to both processes to optimise the plans they find, the relative quality of the plans produced and the distances of the plans from the input plan. We show the best distance and plan quality across *all* plans produced in the entire optimisation phase.[2] All the tests have been performed using 5 minutes of cpu time.

In the first four figures the data is ordered according to the variation scheme we describe above, while in the latter two figures the ordering is based on base problem size.

Speed of Plan Production If we first consider the time to produce a solution (shown in the leftmost graph in each case, using a logscale), we observe that the adaptation process is at least as fast as replanning and, in all cases, taken across the entire data sets the repair strategy is faster than the replanning strategy. In individual cases, for some variants, the replanning approach is faster than repair. This can happen when the modification required to repair a plan has to achieve significant changes in order to support actions that

[2]The first plan generated by LPG, the best quality plan and the best distance plan could be different plans.

are affected by changes to the initial state (it occurs most often in variants with many initial state modifications). In such cases, the work involved in repairing a plan can be greater than in building an initial plan from scratch, but the benefits for investing this effort can be seen in the stability of the plans. It can also be seen that, where smaller problems are used as the base problems for ZenoTravel and Rovers, the times vary far less. This is unsurprising, since where the problem is easier to solve the advantage in working from an existing solution is reduced. Furthermore, the modifications can represent a more significant distortion of the initial state or goals in the case of a smaller problem, so that the original plan offers a less useful guide to the solution of the new problem. Nevertheless, the results show that plan repair outperforms replanning in these cases, too.

Plan Quality We now consider relative plan quality, shown in the rightmost graph in each of these figures. Here we can see a more mixed picture. In figures 1, 4 and 5 we see a marked improvement in quality where plan repair is used. In the latter case, the improvements are only apparent in the larger problems, where there is sufficient complexity in the problem to support alternative solutions. In the case shown in figures 2 the quality is similar using either repair or replanning. In 3 we see an example in which quality is traded off for stability. In contrast to the previous cases, in figures 2 and 3 the base plan was a plan produced by FF. As can be seen, the lower quality of the initial plan appears to reduce the opportunity for repair to exploit its input to achieve a better quality plan for the dynamic problem. The last collection, shown in figure 6, shows virtually no differences in plan quality between repaired or replanned plans. A careful examination of the data reveals a small advantage to the repair strategy, but the smaller problems offer little scope for repair to outperform replanning.

Overall, we see that the repair strategy produces high quality plans, so the exploitation of an existing plan does not undermine the planning process in achieving high quality solutions. Furthermore, in many cases a high quality input plan allows repair to exploit the reasoning already performed to generate this initial plan and to find a higher quality solution to the modified problem instance than can be found by replanning (using LPG to construct the plan).

Plan Stability The central graph in each of these figures shows the relative distances between the original input plan and the new plan produced by repair and by replanning for each variant. The difference in these distances is very marked. Replanning generates plans that are consistently very different to the original input plan. In general, as little as 15-20% of the original plan appears in the replanned plan, while only small proportions of the repaired plans are not common with the original plan. The distinctive saw-cut shapes of the graphs in figures 1, 2, 3 and 4 reflect the scheme we used for generating the variants. Recalling that the first climbing segment of each of these graphs corresponds to a sequence of increasing changes in the initial state, with no changes to the goals, it is clear that the impact of the variations in the initial state is greater than the impact of the changes in goals. This observation is interesting, since it demonstrates that regular minor modifications to the goals (including both deletion of existing goals and introduction of new ones) can be handled very effectively by plan repair, maintaining a very high degree of plan stability. It is also evident that, though the impact of initial state changes is relatively more significant, it is still possible to achieve high degrees of stability when the changes are limited.

Figures 5 and 6 show that higher stability is also achieved by using plan repair in these cases, too. However, where the problems are smaller there is less opportunity for the repair strategy to do better than replanning — in small problems there is often only one way to solve the problem efficiently and this is found by both replanning and plan repair. Here, plan repair is unable to exploit the solution to the original problem, because relatively small numbers of modified initial state or goal literals represent very significant changes in the structure of small problems.

The Impact of Input Plan Quality It seems likely that the quality of the original plan should have some impact on the performance of the plan repair strategy. We explore this impact in figures 7 and 8. In the second of these figures we show speed of plan production when repairing input plans generated by SHOP2 (Nau *et al.* 2003), TALplanner (Kvarnström & Magnusson 2003) and a scheduled version of a plan generated by Metric-FF (Hoffmann 2003). We also show the speed of planning by replanning. In all cases, the repair process is more efficient than replanning, but the times for repair are dependent on the quality of the initial input plan (the TALplanner plan was of highest quality and the SHOP2 plan was of lowest quality). In figure 7 we show both the distances of repaired and replanned plans from the input plan (top row) and the qualities of repaired and replanned plans (bottom row), for the Metric-FF (left), SHOP2 (centre) and TALplanner (right) input plans. Plan stability is not greatly affected by the source or quality of the input plan, but, in general, the higher the quality of the input plan, the higher the quality of the repaired plans. This indicates that the approach we adopt to plan repair achieves a high plan stability and does so while exploiting the best aspects of the input plan. This serves to reemphasise that discarding the input plan, when it might be the product of significant reasoning, is a poor strategy for handling dynamic planning problems.

Input Plans from the Same Planner We have also explored the possibility that stability is enhanced if the input plans are generated by the same planner used for replanning. It might be anticipated that using the same planning strategy for both planning and replanning could lead to smaller plan differences. In figure 9 we show the planning time and the relative distances when using input plans generated by LPG. In the centre graph we show behaviour for input plans generated using LPG tuned for quality and on the right for LPG tuned for speed. In the large majority of the dynamic problems considered for this experiment, the speed of plan repair is not significantly affected by the source of the plans and, as before, overall plan repair produces high quality plans, with greater stability, faster than is achieved by replanning.

As before, the plan distances are better for higher quality input plans. A important observation supported by these

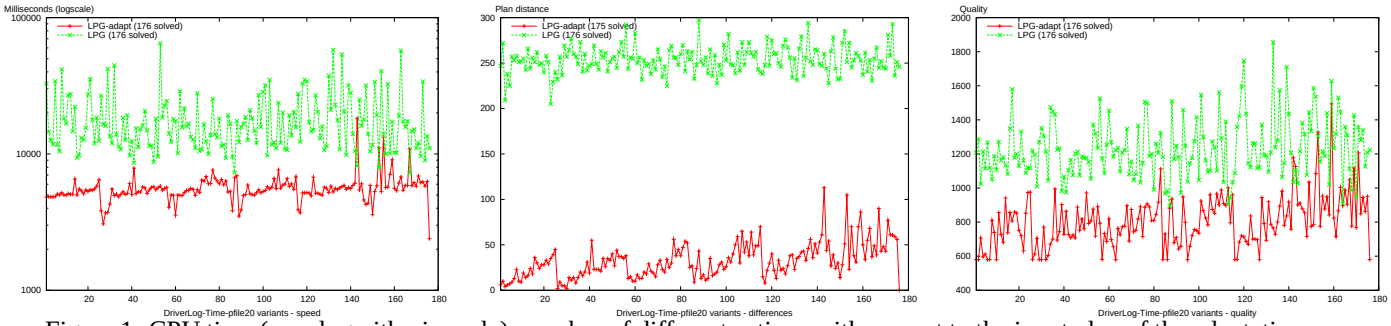

Figure 1: CPU time (on a logarithmic scale), number of different actions with respect to the input plan of the adaptation process and plan qualities for the DriverLog Time-pfile20 variants (considering the median value over 5 runs).

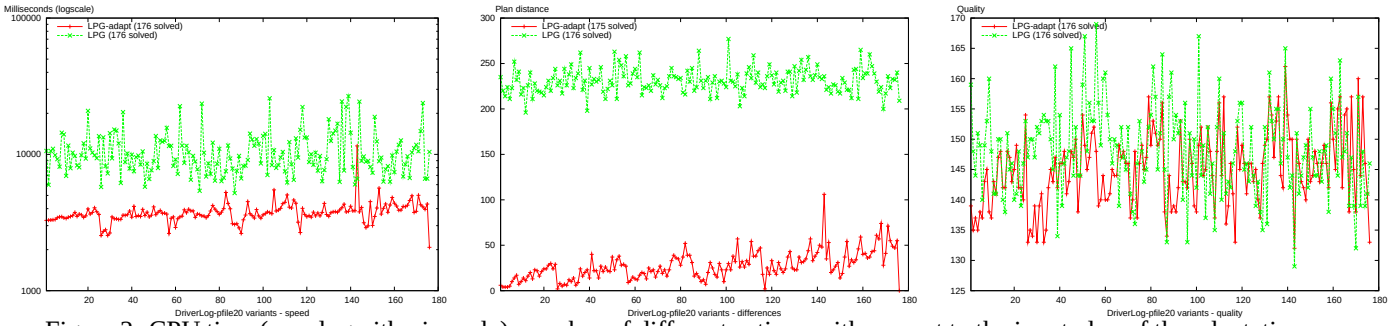

Figure 2: CPU time (on a logarithmic scale), number of different actions with respect to the input plan of the adaptation process and plan qualities for the DriverLog Strips-pfile20 variants (considering the median value over 5 runs).

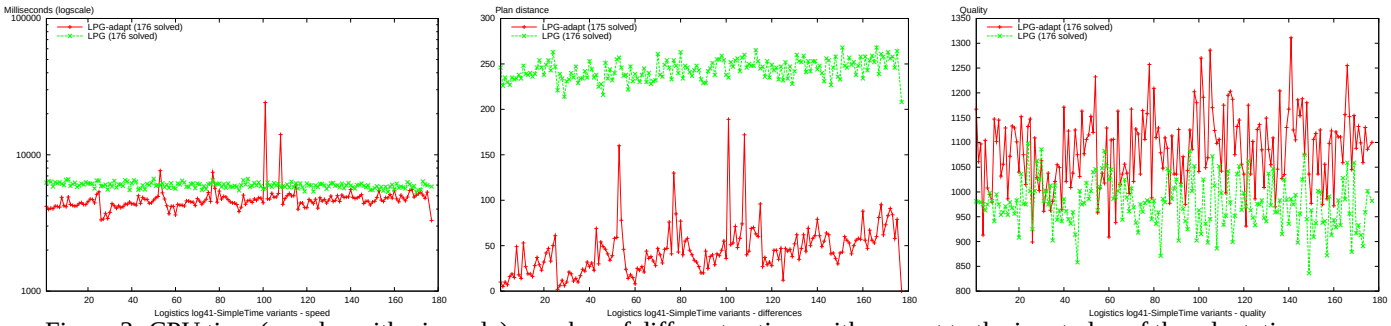

Figure 3: CPU time (on a logarithmic scale), number of different actions with respect to the input plan of the adaptation process and plan qualities for the Logistics log41 Simple Time variants (considering the median value over 5 runs).

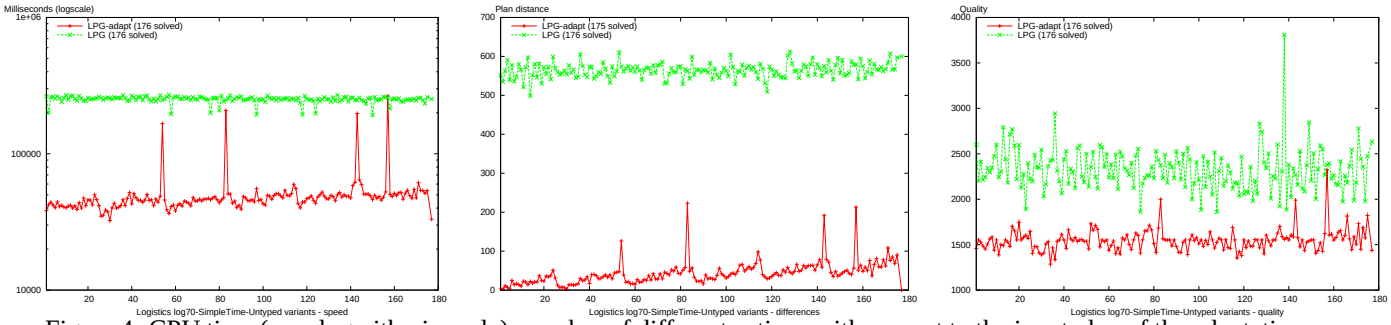

Figure 4: CPU time (on a logarithmic scale), number of different actions with respect to the input plan of the adaptation process and plan qualities for the Logistics log70 SimpleTime Untyped variants (considering the median value over 5 runs).

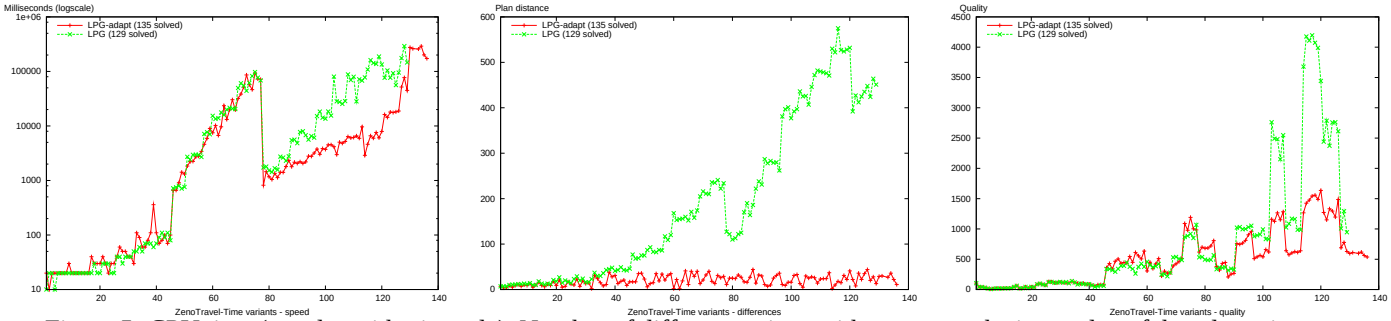

Figure 5: CPU time (on a logarithmic scale), Number of different actions with respect to the input plan of the adaptation process and plan qualities for the ZenoTravel Time variants (considering the median value over 5 runs).

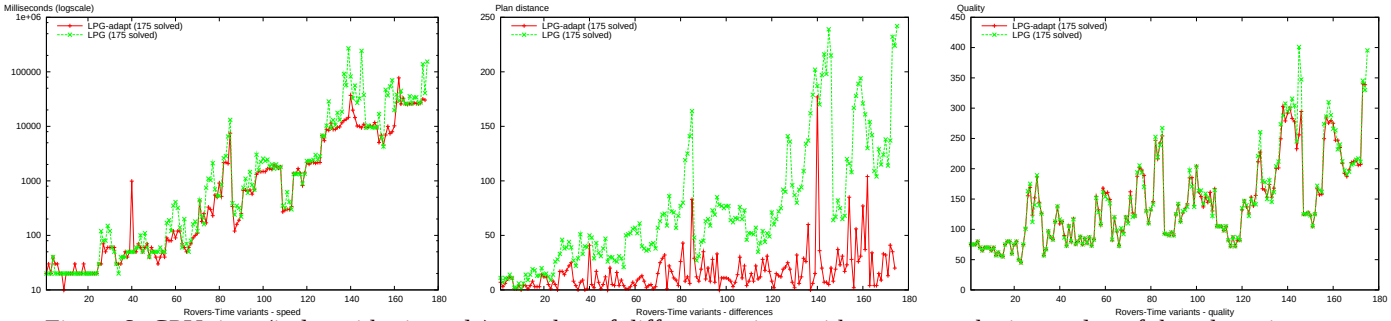

Figure 6: CPU time (in logarithmic scale), number of different actions with respect to the input plan of the adaptation process and plan qualities for the Rover Time variants (considering the median value over 5 runs).

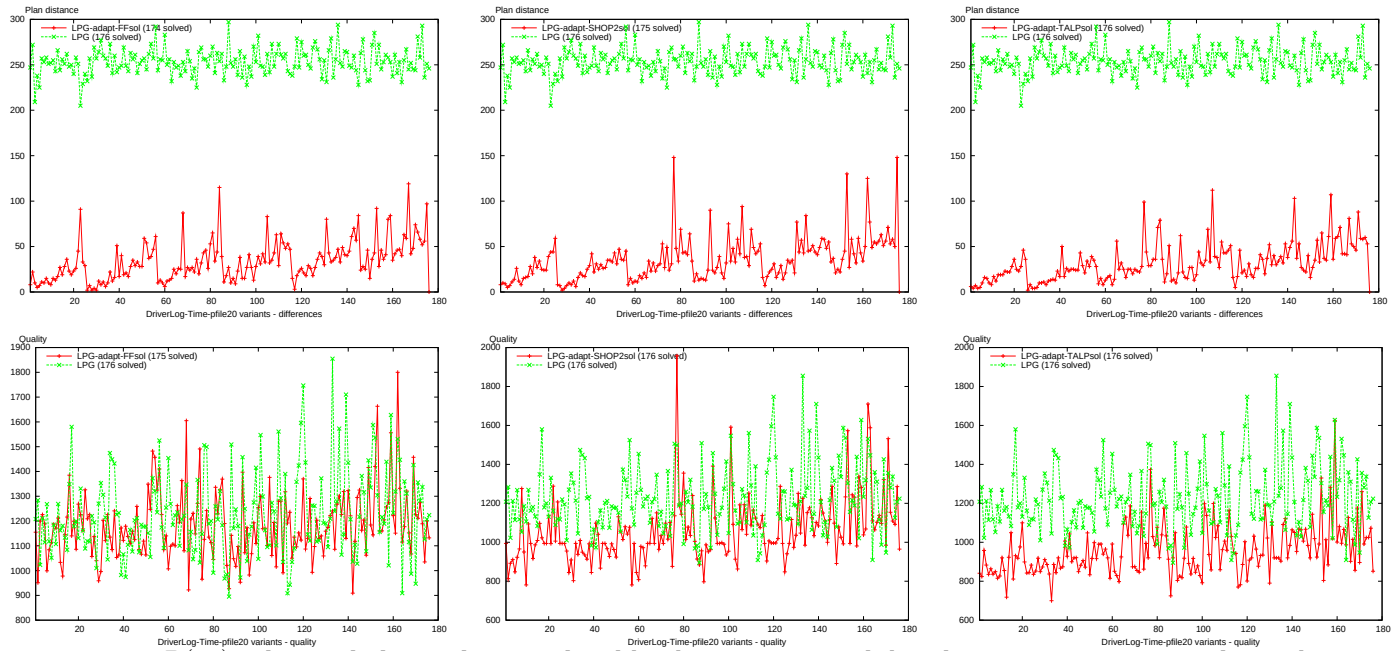

Figure 7: $D(\pi_I)$ values and plan qualities produced by the generation and the adaptation processes considering the input plans generated by FF, SHOP2, TALplanner (plans generated in the 3rd international competition) for DriverLog Time-pfile20 variants (considering the median value over 5 runs).

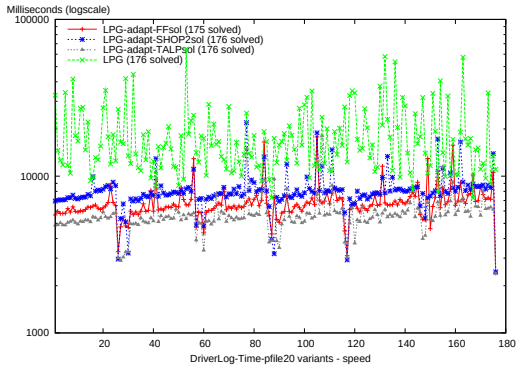

Figure 8: CPU time for replanning and repair using input plans produced by FF, SHOP2, TALplanner for DriverLog Time-pfile20 variants.

results is that the distances for generated plans remain consistent with earlier results, showing that, for LPG at least, the stability is not significantly improved by using the same planner to replan as was used to generate the original plan. It is possible that this result would not hold to the same degree if a *deterministic* planner were used for the comparison.

Plan Repair for Rapid Response We claim that an advantage offered by repair is that, by exploiting work already performed in constructing the original input plan, a high quality plan can be constructed very quickly. By simply confirming that the changed context does not invalidate the parts of the plan it maintains, the repair strategy does not have to repeat the reasoning that led to the construction of this plan in the first place, but can still benefit from it. One situation that can arise when solving dynamic planning problems is that a change in context occurs very shortly before a decision must be made whether to commit to the next action or not. In this case, producing a high quality plan very fast becomes far more important, with the opportunity to optimise a plan using extra time becoming less useful.

In figure 10 we show that the benefits of using a good quality input plan are apparent in the very first output solution from LPG, using the repair process. The upper graph shows that the stability for first solutions follows the same pattern as for solutions following optimisation. This confirms that the repair approach does not quickly generate a new solution and then optimise it towards the original plan, but rather it exploits the original solution from the outset. The lower graph shows that the quality of the first repaired plan is also much higher than the first solution generated by replanning. This confirms that, from the outset, the emphasis on stability does not compromise quality and that the original high quality solution provides a direct and powerful guide to solving the new problem instance.

An alternative way to examine the speed of reaction to changing contexts is to consider how often the first step of the original plan remains a first step of the new plan. If we can show that plan repair frequently leaves the first step stable then we can exploit a powerful heuristic in order to spend longer finding a new plan in reaction to a change in context: check whether the next step of the original plan is still executable and, if so, execute it and spend all the time up to the end of this action repairing the plan for execution from this point. In domains where actions take significant time to execute compared with time to plan, this can be a valuable opportunity to improve plan quality. The following table shows our results for 880 DriverLog-Time variants:

Strategy	Init	Orig	Diff	Eq 1st
Repair	9517	75.8%	8.9%	100%
Replan	5645	47.7%	66.0%	44.3%

The columns show: the number of initial actions over all the plan (there are typically many parallel initial actions), the percentages of these that appear in the original collection of initial actions (Orig) and the percentages of the original initial actions that are not included in the new plans (Diff). The final column (Eq 1st) shows the percentages of plans whose first listed action matches the first listed action of the original plan. These figures strongly support our claim that the repair strategy can be coupled with the proposed heuristic.

6 Conclusions

We have explored the problem of reacting to a dynamic environment, where plans fail because of changing contexts. We have compared a repair-based approach with the alternative of constructing a new plan *ab initio*. An important motivation for choosing to repair a plan is that plans provide a means to communicate future intentions to other agents and, to do so successfully, it is important to ensure that, as contexts shift and plans become invalid, new plans are not unnecessarily different from existing plans. We define a concept of plan stability to reflect the degree of change a repair process induces in a plan.

We have used LPG as our basis for performing plan repair, because it already uses a strategy that can be closely compared to plan repair. LPG performs an iterative cycle of candidate revision in a local search process. By modifying the evaluation function, we have adapted LPG to emphasise the importance of plan stability. This is achieved by penalising changes that drift away from the original input plan, while still accounting for the need to find a solution and to maintain plan quality and temporal compactness.

We have shown, empirically, that this modified LPG achieves good results in a plan repair strategy, producing plans more efficiently and of better quality than by applying LPG to construct a plan as if for a new problem. Furthermore, the approach supports the discovery of plans with greater stability. This demonstrates that, even for these benchmark problems, there is a wide variation in the plans that are possible, so that an uninformed search for a plan will be unlikely to produce a plan that is close to a previous plan for a similar problem. In other words, there are dynamic planning problems with closely similar initial state and goals, but for which there are (good) plans at a considerable distance from the original plan. We believe that many useful dynamic planning problems are like this: that is, the solution space for many planning problems is underconstrained, with many different ways to solve them. In

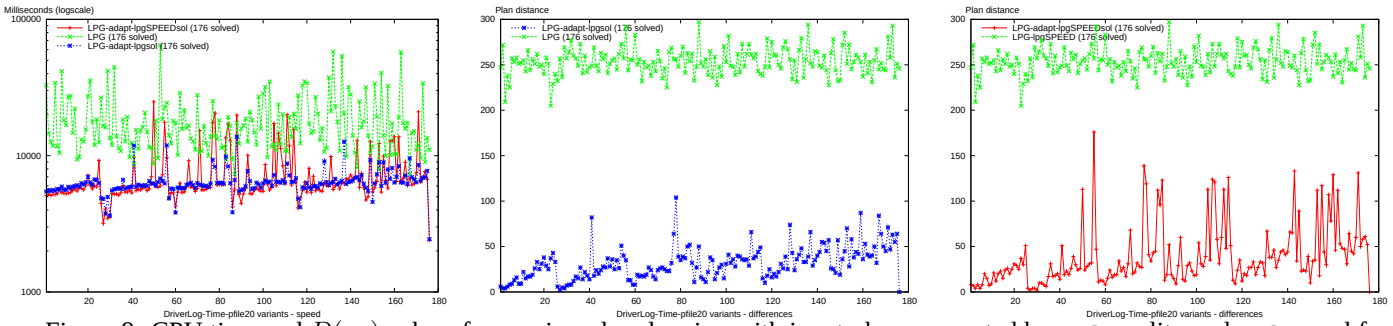

Figure 9: CPU time and $D(\pi_I)$ values for repair and replanning with input plans generated by LPG-quality and LPG-speed for DriverLog Time-pfile20 variants. median value over 5 runs.

these cases, and when plan stability is important, plan repair can offer a far more useful strategy than replanning.

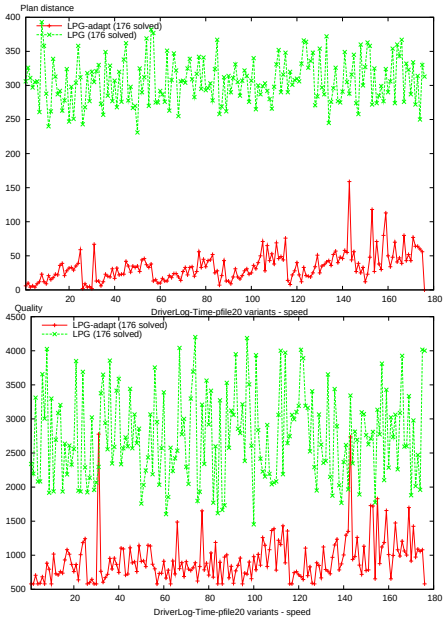

Figure 10: Plan distance and plan qualities for the DriverLog Time-pfile20 variants considering the first solution generated (median value over 5 runs).

References

Bacchus, F., and Kabanza, F. 2000. Using temporal logic to express search control knowledge for planning. *Artificial Intelligence* 116(1-2):123–191.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *J. AI Res.* 11:1–94.

Cushing, W., and Kambhampati, S. 2005. Replanning: a new perspective. In *Proc. of ICAPS*.

Drummond, M.; Bresina, J.; and Swanson, K. 1994. Just-in-case scheduling. In *Proc. 12th Nat. Conf. on AI (AAAI)*, 1098–1004.

Frank, J.; Gross, M. A. K.; and Kürklü, E. 2004. SOFIA's choice: An AI approach to scheduling airborne astronomy observations. In *Proc. of AAAI*, 828–835.

Gerevini, A., and Serina, I. 2000. Fast plan adaptation through planning graphs: local and systematic search techniques. In *Proc. of Int. Conf. on AI Planning and Scheduling (AIPS'00)*.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs. *J. AI Res.* 20.

Hammond, K. 1990. Explaining and repairing plans that fail. *Artificial Intelligence* 45:173–228.

Hanks, S., and Weld, D. 1995. A domain-independent algorithm for plan adaptation. *J. AI Res.* 2:319–360.

Hoffmann, J. 2003. The Metric-FF planning system: Translating "ignoring delete lists" to numerical state variables. *J. AI Res.* 20.

Horty, J. F., and Pollack, M. E. 2001. Evaluating new options in the context of existing plans. *Artificial Intelligence* 127:199–220.

Kambhampati, S., and Hendler, J. 1992. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence* 55:193–258.

Kambhampati, S. 1990. Mapping and retrieval during plan reuse: a validation structure based approach. In *Proc. of AAAI*, 170–175.

Koenig, S.; Furcy, D.; and Bauer, C. 2002. Heuristic search-based replanning. In *Proc. of Int. Conf. on AI Planning and Scheduling (AIPS'02)*, 310–317.

Kvarnström, J., and Magnusson, M. 2003. Talplanner in the 3rd international planning competition: Extensions and control rules. *J. AI Research* 20.

Nau, D.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdoch, J.; Wu, D.; and Yaman, F. 2003. An HTN planning environment. *J. AI Res.* 20.

Nebel, B., and Koehler, J. 1995. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence* 76(1-2):427–454.

Pryor, L., and Collins, G. 1996. Planning for contingencies: A decision-based approach. *J. AI Res.* 4:287–339.

Simmons, R. 1988. A theory of debugging plans and interpretations. In *Proc. of AAAI*, 94–99.

Stulp, F., and Beetz, M. 2005. Optimized execution of action chains using learned performance models of abstract actions. In *Proc. of IJCAI*, 1272–1278.

Traum, D., and Allen, J. 1994. Towards a formal theory of repair in plan execution and plan recognition. In *Proc. UK PlanSIG*.

van der Krogt, R., and de Weerdt, M. 2005. Plan repair as an extension of planning. In *Proc. of the 15th Int. Conf. on Automated Planning and Scheduling (ICAPS-05)*, 161–170.

Veloso, M. 1994. *Planning and learning by analogical reasoning*. LNCS 886. Springer-Verlag.