Computing Opportunities to Augment Plans for Novel Replanning during Execution

Daniel Borrajo[*] and **Manuela Veloso**[†]
J.P.Morgan AI Research, New York, NY (USA)
{daniel.borrajo,manuela.veloso}@jpmchase.com

Abstract

Traditionally, planning provides for execution plans as sequences of actions with preconditions and effects. Execution monitoring identifies failure conditions when the preconditions of an action do not match the state. Interestingly, planning proceeds by consuming a given initial state and abandoning reasoning about any facts not true in that state. In this paper, we define opportunities as such missing facts, and contribute an algorithm to compute them and augment a plan for execution with them. We then introduce a new execution opportunity monitoring that focusedly checks for these opportunities at each execution state. Opportunistic replanning proceeds now from the new state including the detected opportunities.

Introduction

Planning deals with the task of generating sequences of actions, plans, to be later executed. The execution system applies each action in the sequence and also monitors the new states. When executing plans in dynamic environments, the monitoring system checks whether the state after executing any action matches the one expected from the domain model. This usually entails checking if preconditions of future actions still hold. If there is a mismatch, a replanning system generates a new plan from the current state. We will call this approach *replanning-F*, as replanning from failure (Yoon, Fern, and Givan 2007).

A key observation is that planners focus on the given initial state. Therefore, they do not reason about alternative scenarios. Some current planners even remove from the search space those actions whose preconditions are static facts and do not hold in the initial state (Richter and Westphal 2010). However, in dynamic environments these facts could become true during execution. So, if planners would reason about them, they could find better plans than the one in execution. In this paper, we define such facts as *opportunities*. If they are found during execution, we advocate for a second type of replanning, *replanning-O*, as replanning from opportunities. The execution will change the plan in execution if the plan taking into account the opportunities is better than the remaining part of the current plan. This can be seen as a special case of counterfactual reasoning (Pearl 2013; Silva, Melo, and Veloso 2018).

Example. *Let us present the* DOCUMENTS *domain. Suppose a robot needs to hold some documents. They are stored inside a briefcase that it carries, but initially it does not have the key of the briefcase. Fortunately, there is a copy of each document, but they are distributed across several rooms. So, the initial plan is to* `move` *around rooms to* `grab` *the copies of the documents. Now, suppose that at any time the robot finds the key. Since it is not a failure of the plan in execution, replanning-F monitoring systems would continue with the execution of the current plan. However, the key (opportunity) allows to compute a better plan by opening the briefcase with the key, avoiding to move around offices (using the* `grab-with-key` *action). So, in replanning-O, the planning module would first notice while searching that having the key might be relevant to generate a new plan. Then, the monitoring module could check at each action execution step whether it has the key. If so, it would replan; not because it is a failure of the current plan, but because it found an opportunity to generate a better plan. The briefcase might be empty, so it has to replan to see if there is a better plan. If a better plan is found, it can shift to the new plan.*

Many real-world domains have opportunities as defined here. Finding an item in the current shop, discovering a gas station while driving, getting access to a new database, or someone letting us use a tool instead of buying it.

The main contribution of this paper is a domain- and planner-independent technique to automatically: extract opportunities when planning; augment the plan with them; and monitor the action execution for performing replanning-O if appropriate. The paper also includes experiments as an illustration on the benefits it can provide to plans execution.

Background

A classical planning task is a tuple $\Pi = \{F, A, I, G\}$, where F is a set of propositions, A is a set of instantiated actions, $I \subseteq F$ is an initial state, and $G \subseteq F$ is a set of goals. Each action $a \in A$ is described by a set of preconditions (pre(a)) and a set of effects (eff(a)). We assume the standard seman-

[*]On leave from Universidad Carlos III de Madrid (consultant)
[†]On leave from Carnegie Mellon University

tics for preconditions, effects and applicability of actions. Each action might incur a cost, $c(a)$. The solution to a planning task is a plan; a sequence of actions $\pi = (a_1, \ldots, a_n)$ such that if applied in order from the initial state I would result in a state s_n, where goals are true, $G \subseteq s_n$. Plan cost is commonly defined as: $C(\pi) = \sum_{a_i \in \pi} c(a_i)$. Now, we define general opportunity and Plan-Based opportunities.

Definition 1 (General opportunity)**.** *Given a planning task $\Pi = \{F, A, I, G\}$, a fact $o \in F$ is an opportunity if: (i) $o \notin I$; and (ii) it is static (no action can add or delete it, $\nexists a_i \in A, o \in$ eff(a)).*

Definition 2 (Plan-Based opportunity)**.** *Given a planning task $\Pi = \{F, A, I, G\}$, and a plan π that solves Π, an opportunity $o \in F$ is a Plan-Based opportunity if:*
$\exists a \in A, a \notin \pi, o \in$ pre$(a), \exists a' \in \pi, e \in$ eff$(a) \cap$ eff(a').

A Plan-Based opportunity is a static precondition of an action a not in the plan that achieves an effect that is also achieved by some other action a' in the plan. So, if o would be true, the planner could use a instead of a' in the plan.

Computation of Opportunities

A way to obtain opportunities would be to compute them while the planner searches for a solution. We could store as opportunities the static literals that were checked against the state and did not hold. This is easy to obtain from old backward-chaining planners (Veloso et al. 1995; Penberthy and Weld 1992), but hard to obtain from current planners. Modern planners prune actions at the pre-processing step by removing ground actions if they have as a precondition some fact that is static and it is not true in I.

Plan-based Opportunities

PLAN-BASED-OPPORTUNITIES, PBO, (Algorithm 1) first regresses the goals through the plan. Then, it analyzes the actions that achieve any goal in the goal regression, and the planner did not reason about them because they were not part of the action set; they had at least one precondition that is a static fact that was not true in I. Those preconditions would be the opportunities.

The algorithm traverses the plan from the end backwards in order to perform goal regression (step 3). For each goal g in the regressed goals, if it is true in the initial state, the algorithm removes the goal from the regressed goals (step 6). Otherwise, it iterates over all grounded actions ga that could achieve g (function ACHIEVERS(A, g) returns that set). If ga was the one selected in that step of the plan (step 10), it adds its preconditions to the regressed goals set. Otherwise, all the static preconditions of ga that were not true in the initial state are added to the opportunities of that action.

PBO stores opportunities with each action a in the plan, $O(a)$. Thus, when some action a in the plan is later executed, monitoring only considers opportunities $o \in O(a)$. This definition of opportunities is quite restrictive, since replanning-O will only be considered when some opportunity in $O(a)$ appears precisely when executing a. In order to make it more general, we propagate opportunities back through the plan (using function PROPAGATE-OPPORTUNITIES) by adding each opportunity in $O(a_i)$ to $O(a_k), k = 1..i-1, \forall a_k \in \pi$. A key observation is that PBO is planner-independent, so we do not have to modify the code of existing planners to compute the opportunities. Also, PBO computes a bounded set of opportunities that are directly relevant to the plan.

Algorithm 1 PLAN-BASED-OPPORTUNITIES(Π, π)

Inputs: planning task $\Pi = \{F, A, I, G\}$, plan π
Outputs: opportunities vector $O(a)$ $(\forall a \in \pi)$

1: $O \leftarrow \langle \emptyset, \emptyset, \ldots, \emptyset \rangle$ $\quad |O| = |\pi|$
2: regressed$\leftarrow G$
3: $\pi' \leftarrow$REVERSE(π)
4: **while** π' not empty **do**
5: **for all** $g \in$regressed **do**
6: **if** $g \in I$ **then**
7: regressed$\leftarrow$regressed$\backslash\{g\}$
8: **else**
9: **for all** $ga \in$ACHIEVERS(A, g) **do**
10: **if** $ga =$ FIRST(π') **then**
11: regressed$\leftarrow$REGRESS(regressed,pre(ga))
12: **else**
13: **for all** $p \in$ pre(ga) **do**
14: **if** $p \notin I$ AND p is static **then**
15: $O(ga) \leftarrow O(ga) \cup \{p\}$
16: POP(π')
17: $O \leftarrow$PROPAGATE-OPPORTUNITIES(O, π)
18: **return** O

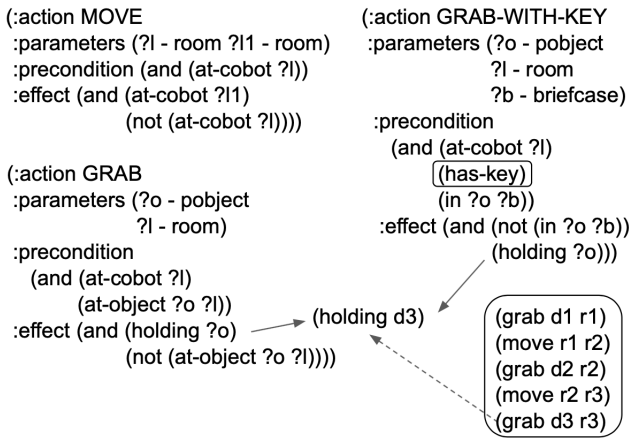


Figure 1: Three actions in the DOCUMENTS domain, a plan and how the opportunity `(has-key)` is computed.

Example. *Figure 1 shows an example of computing opportunities in a problem in the* DOCUMENTS *domain. There are three rooms (r1 to r3), each room with a copy of one of the three documents (d1 to d3) inside the briefcase. The goal is to hold the three documents (or their copies). The plan is in the bottom right of the figure. The last step achieves the goal* `(holding d3)` *by using the* `grab` *action. Another action in the domain can achieve the same goal,* `grab-with-key`*. However, it could not be used because one of its preconditions,* `(has-key)`*, is not true in the*

Algorithm 2 OPPORTUNITIESPLANEXECUTION(Π)

Inputs: planning task $\Pi = \{F, A, I, G\}$
Outputs: boolean (solved or not)

$\pi \leftarrow$PLAN(Π)
$O \leftarrow$PLAN-BASED-OPPORTUNITIES(Π, π)
$s \leftarrow I$
while π not empty AND $G \not\subseteq s$ **do**
  $a \leftarrow$POP(π)
  $s' \leftarrow$EXECUTE(a, Π, O)
  if NEEDREPLANNING(s', π, a, O) **then**
    $\pi', O \leftarrow$PLANOPPORTUNITIES($\{F, A, s', G\}, \pi, O$)
    if π' not empty **then**
      $\pi \leftarrow \pi'$
  $s \leftarrow s'$
if $G \subseteq s$ **then**
  return True
else
  return False

Algorithm 3 PLANOPPORTUNITIES(Π, π, O)

Inputs: planning task $\Pi = \{F, A, s', G\}$, current plan π, opportunities O
Outputs: plan π, opportunities O

$\pi' \leftarrow$PLAN(Π)
if π' not empty AND $C(\pi') < C(\pi)$ **then**
  $\pi \leftarrow \pi'$
  $O \leftarrow$PLAN-BASED-OPPORTUNITIES(Π, π')
return π, O

Initial plan	New plan
(grab d1 r1)	
(move r1 r2)	(grab-with-key d2)
(grab d2 r2)	(grab-with-key d3)
(move r2 r3)	
(grab d3 r3)	

Table 1: Example of execution where the key is found before executing the second step in the initial plan (left). Then, it changes to the new remaining plan (right).

initial state. Algorithm 1 would find that alternative and it would define `(has-key)` *as the only opportunity.*

Replanning

Algorithm 2 redefines a planning-execution cycle. It takes as input a planning task and returns a boolean indicating whether the planning-execution cycle achieved the goals or not. The algorithm first computes a plan through function PLAN. Then, it computes opportunities based on that plan according to Algorithm 1.

EXECUTE executes the action and returns the new sensed state. We have implemented an environment that adds opportunities.[1] In the case of PBO, the system only needs to know whether any of the pre-computed opportunities changed its truth value. So, there is no need to sense the complete state; only, the truth value of those opportunities. We do not consider replanning-F as it is orthogonal to this work. In case PBO is combined with replanning-F, it would check for the opportunities plus the preconditions that are part of the goal regression. NEEDREPLANNING checks whether it has found an opportunity. In that case, it replans and checks whether it should change the remaining part of the plan in execution.

Algorithm 3 describes PLANOPPORTUNITIES that computes a plan from the new state. If the new plan's cost is better than the previous one, it computes a new set of opportunities, and returns the new plan and set of opportunities.

Example. *Table 1 shows what happens if the robot finds the key before executing the second step. It would replan, and find that the new plan (right column) is better than the initial one (left column), so it changes to the new plan.*

Illustrative Example

This section presents an example of how the algorithm works. We compare the PBO algorithm against a baseline, that replans when there is any change in the environment, Replan When Change (RWC). In this environment, the robot sees one new object everytime it executes an action. As explained before, PBO only replans when it finds an opportunity computed with Algorithm 1. There is also a big difference on how these algorithms sense the environment. RWC would sense all the new state, while PBO would only sense whether any fact in O changed its truth value. We varied the step at which the opportunity becomes true to analyze the effect of finding it sooner or later, using two schemes: fixed (at steps 1, 5 and 10); and proportional (10%, 50%, and 90% of the original plan length).

We used as planner LAMA-FIRST, the first iteration of LAMA (Greedy best-first search with a combination of the FF and the landmark count heuristics) (Richter and Westphal 2010) implemented in Fast Downward (Helmert 2006).

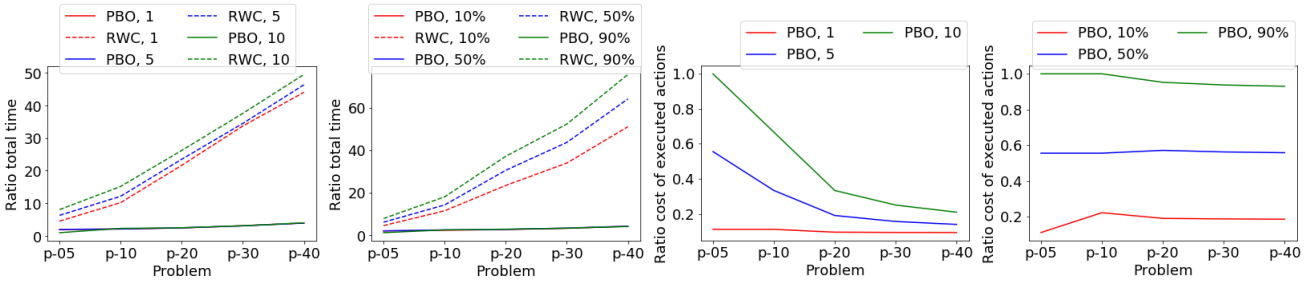
The DOCUMENTS domain has been previously defined in this paper. The `grab` actions have a cost of 1, and the `move` action has a cost of 10. We generated problems with increasing number of documents and rooms from 5 to 40. Experiments were performed on a MacBook Pro, with processor 2.4 GHz Intel Core i5, 4Gb of RAM. We set a time bound for the complete planning-execution cycle of 1800s, a time bound for each planning episode of 500s and a time bound for each time it computes opportunities of 300s.

Results

Figures 2(a) and (b) show the ratio between the total time and the initial planning time for PBO and RWC. (a) shows the fixed steps and (b) the proportional steps. The problems are on the x-axis, while the different lines correspond to the different values of the steps where an opportunity appears. Similarly, Figures 2(c) and (d) show the ratio between the total cost of the executed actions by PBO and the initial plan cost. Smaller values are better.

Time to solve. The ratio of the total time PBO spends with respect to the initial planning time is an order of magnitude

[1] Through the EXECUTE function.



x-axis: Problem, y-axis: Ratio total time (a) (b)

x-axis: Problem, y-axis: Ratio cost of executed actions (c) (d)

Figure 2: Ratio between total and initial planning times for (a) fixed and (b) proportional appearance of opportunities. Ratio between the total cost of the executed actions and the initial plan cost for (c) fixed and (d) proportional.

less than RWC even if PBO has to add the time to compute the opportunities. The main differences between PBO and RWC are: (1) PBO performs focused monitoring, so it will only replan if an opportunity appeared; and (2) once all possible opportunities have appeared, PBO does not replan again. Instead, RWC replans at each time step something changes. If generating a plan is expensive, RWC spends a huge amount of time replanning. Also, there is very little influence on the computation time of the steps where the opportunity arises.

Execution cost. The difference in execution cost for PBO and RWC is negligible, so we only show the values for PBO. When the opportunity appears early (fixed), it gets a reduction on 80-90% on the cost independently on how difficult the problem is. The later it appears, the benefit on cost increases with the difficulty of the problem for fixed (Figure 2(c)). For proportional (d), the cost ratio is proportional to 10%, 50% and 90% with slight variations due to odd values of these percentages and the different cost of actions.

Related Work

Opportunities have been studied in cognitive science and early work in planning (Birnbaum and Collins 1984; Seifert and Patalano 2001; Schank and Abelson 1977). In cognitive science, the definition of opportunities was domain-dependent and performed by humans. The driver for opportunities was the pending goals that were associated with cues. In our case, opportunities come from goal regression.

Recent approaches have used different definitions of opportunities: pre-defined goals that could be achieved in case of underused resources or resources uncertainty (Cashmore et al. 2017; Coles 2012; Horvitz, Koch, and Subramani 2007); new goals to be pursued (with little or no re-planning capabilities) (Lawton and Domshlak 2004); or visiting some specific waypoints in robotics applications (Thakur et al. 2013). Other replan for new goals requests (Haigh and Veloso 1998) or to anticipate new goals (Fuentetaja, Borrajo, and de la Rosa 2018). Our approach is domain-independent and defines opportunities as state facts, not goals.

A related task consists of explaining why a plan failed (Göbelbecker et al. 2010; Klenk, Molineaux, and Aha 2013). These explanations can yield changes to the goals, state or actions. The algorithms used to compute those changes are able to capture the relevant facts from analysis of the planning task similarly to our algorithms. However, they: only compute excuses when the plan failed; use the causal and domain transition graphs instead of doing plan-based goal regression; and do not have a plan-execution loop that considers the opportunities to improve execution.

Others have challenged assumptions made during planning and planned with different levels of detail (Trevizan and Veloso 2014; Zickler and Veloso 2010; Martínez, Fernández, and Borrajo 2016), learned to better match the environment (Jiménez, Fernández, and Borrajo 2013), performed rationale-based monitoring (Veloso, Pollack, and Cox 1998) or checked for optimality (Fritz and McIlraith 2007).

Conclusions

The main contributions of this paper are algorithms to compute opportunities, and to focusedly monitor for those opportunities. We show through experiments in an illustrative domain that opportunities can yield to significant improvements on planning-execution time over replanning when something changes in the environment, and cost/length of executed plans over no replanning.

Acknowledgements

This paper was prepared for information purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co. and its affiliates ("JP Morgan"), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful. This work has been partially funded by FEDER/Ministerio de Ciencia, Innovación y Universidades - Agencia Estatal de Investigación, TIN2017-88476-C2-2-R.

References

Birnbaum, L.; and Collins, G. 1984. Opportunistic planning and Freudian slips. In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, 124–127.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; and Ridder, B. 2017. Opportunistic Planning in Autonomous Underwater Missions. *IEEE Transactions on Automation Science and Engineering* 15(2): 519–530.

Coles, A. 2012. Opportunistic Branched Plans to Maximise Utility in the Presence of Resource Uncertainty. In *Proceedings of ECAI*, 252–257.

Fritz, C.; and McIlraith, S. A. 2007. Monitoring Plan Optimality During Execution. In *Proceedings of ICAPS*, 144–151.

Fuentetaja, R.; Borrajo, D.; and de la Rosa, T. 2018. Anticipation of Goals in Automated Planning. *AI Communications* 31(2): 117–135.

Göbelbecker, M.; Keller, T.; Eyerich, P.; Brenner, M.; and Nebel, B. 2010. Coming Up with Good Excuses: What To Do When No Plan Can be Found. In *Proceedings of ICAPS*, 81–88.

Haigh, K. Z.; and Veloso, M. M. 1998. Interleaving Planning and Robot Execution for Asynchronous User Requests. *Autonomous Robots* 5(1): 79–95.

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26: 191–246.

Horvitz, E.; Koch, P.; and Subramani, M. 2007. Mobile Opportunistic Planning: Methods and Models. In *Proceedings of User Modeling Conference*, 228–237.

Jiménez, S.; Fernández, F.; and Borrajo, D. 2013. Integrating Planning, Execution and Learning to Improve Plan Execution. *Computational Intelligence Journal* 29(1): 1–36.

Klenk, M.; Molineaux, M.; and Aha, D. W. 2013. Goal-Driven Autonomy For Responding To Unexpected Events In Strategy Simulations. *Computational Intelligence* 29(2): 187–206.

Lawton, J. H.; and Domshlak, C. 2004. Multi-Agent Opportunistic Planning and Plan Execution. In *Proceedings of ICTAI*, 408–416.

Martínez, M.; Fernández, F.; and Borrajo, D. 2016. Planning and Execution through Variable Resolution Planning. *Robotics and Autonomous Systems* 83: 214–230.

Pearl, J. 2013. Structural Counterfactuals: A Brief Introduction. *Cognitive Science* 37(6): 977–985.

Penberthy, J. S.; and Weld, D. S. 1992. UCPOP: A Sound, complete, partial order planner for ADL. In *Proceedings of KR*, 103–114.

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

Schank, R.; and Abelson, R. 1977. *Scripts, plans, goals and understanding*. Erlbaum.

Seifert, C. M.; and Patalano, A. L. 2001. Opportunism in memory: Preparing for chance encounters. *Current Directions in Psychological Science* 10(6): 198–201.

Silva, R.; Melo, F.; and Veloso, M. 2018. What if the World Were Different: Gradient-Based Exploration for New Optimal Policies. In *Proceedings of General Conference on Artificial Intelligence*, 229–242.

Thakur, D.; Likhachev, M.; Keller, J.; Kumar, V.; Dobrokhodov, V.; Jones, K.; Wurz, J.; and Kaminer, I. 2013. Planning for opportunistic surveillance with multiple robots. In *Proceedings of IROS*, 5750–5757.

Trevizan, F.; and Veloso, M. 2014. Depth-based Short-sighted Stochastic Shortest Path Problems. *Artificial Intelligence Journal* 216: 179–205.

Veloso, M.; Carbonell, J.; Pérez, A.; Borrajo, D.; Fink, E.; and Blythe, J. 1995. Integrating Planning and Learning: The PRODIGY Architecture. *Journal of Experimental and Theoretical AI* 7: 81–120.

Veloso, M. M.; Pollack, M. E.; and Cox, M. T. 1998. Rationale-Based Monitoring for Planning in Dynamic Environments. In *Proceedings of AIPS*, 171–180.

Yoon, S.; Fern, A.; and Givan, R. 2007. FF-Replan: A Baseline for Probabilistic Planning. In *Proceedings of ICAPS*, 352–360.

Zickler, S.; and Veloso, M. 2010. Variable Level-of-Detail Motion Planning in Environments with Poorly Predictable Bodies. In *Proceedings of ECAI*, 189–194.