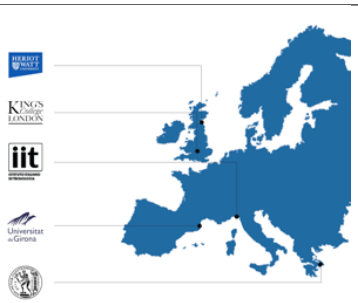
Pandora

Persistently Autonomous Robots

Heriot-Watt University Istituto Italiano di Tecnologia Universitat de Girona King's College London National Technical University of Athens	 www.persistentautonomy.com	Reference Code: Pandora-D2.3

PUBLIC

Persistent Autonomy through Learning, Adaptation, Observation and Replanning

DELIVERABLE 2.3

Opportunistic planning

Distribution:	Pandora Partners, Project Officer, Reviewers, PANDORA website
Supplementary notes:	

Rev.	Date	Description	Pages	Prepared	Checked	Approved	Authorised
5							
4							
3							
2							
1	2015-07-31	Final Version	29	KCL	KCL	KCL	HWU
0	2015-05-30	First Draft Issued	29	KCL	KCL	KCL	KCL

This document is the property of the Pandora partnership. - Any unauthorised use of the same will be prosecuted

Contents

Introduction

The PANDORA project aims to achieve *persistent autonomy*, through planning, task learning, plan execution within resource limits and adaptive response to interesting unanticipated opportunities. The operational context is an underwater oil installation, consisting of manifolds, pipelines, valves and welds, requiring regular inspection and maintenance. The operations are performed by two underwater vehicles, one (Nessie) equipped for inspection only, and the other (Girona-500) equipped for interventions such as valve-turning. Valves must be turned within time windows that are known to the plan generation system. The installation must be maintained over long periods, such as days or weeks, without human intervention. Because of energy and time constraints, the daily operations of the AUVs must be planned in advance, to ensure that best use (measured in terms of the number and utility of tasks completed) is made of limited resources. The situation is complicated by the fact that environmental conditions (such as currents and marine life) might affect how long tasks take to complete, and when they are available for completion. During the execution of a plan by the AUV, unexpected events might occur that provide opportunities for the vehicle to increase the overall utility of its operations. An example is that a part-submerged section of an anchor chain, or other structure, might be spotted during operations. This event provides an opportunity to perform an unplanned inspection, or chain-following activity, provided that resources permit the execution of the necessary extra actions.

We assume that these opportunities are rare, but offer high utility gain when they are spotted and can be exploited. Thus, opportunities in this framework are somewhat similar to *high impact, low probability* events (HILPs) [9]. The probability density function governing their distribution in the physical space is unknown, so we cannot plan to anticipate them or determine their expected utility.

In this work, we present an approach that supports this opportunistic behaviour, leading to better utility than was planned for, without the need for probabilistic modeling or methods. We compare our approach to a reactive replanning strategy. Our results show that opportunistic planning is less resource-intensive than replanning, and leads to higher overall achieved utility. To construct plans, we use the POPF planner [2] and planning domain models written in the temporal planning language PDDL2.1 [5].

Chapter 1

Command and Control in PANDORA

In the military command and control paradigm, strategic planning of a mission is performed by Brigadiers and Corps Commanders, reasoning at the level of units and divisions and their movements, locations and strategic operations. The uncertainties to be encountered at the control level, by platoon and squad commanders, are not addressed at the command level. In PANDORA, the strategic plan will determine the missions (valve operations and inspections) that must be achieved, across the installation, over the planning horizon. A strategic plan allocates resources to these missions and determines the order in which the missions should be achieved to ensure the best use of those resources.

The strategic plan is constructed, in cooperation with *tactical* solvers, and then refined by the tactical solvers whose task is to plan how to achieve the mission components of the strategic plan within the given resource constraints. In building the overall mission plan, the strategic level needs to interact with the tactical level in order to obtain good estimates of resource requirements. During the refinement phase, the tactical level is ignorant of the overall strategic plan: tactical solvers are ignorant of the over-arching mission structure, and only know the resource envelopes in which they must work to achieve their mission components. In PANDORA, the mission components are individual valve-turning and inspection missions.

At the operational level, controllers must work within the resource budgets assigned to them by the tactical and strategic levels. However, they have flexibility about how to interpret their commands within these limits and within the strategic parameters. It is at this level that opportunities might arise, which can then be pursued at the operational level, if resources allow. Planning to exploit these opportunities is, again, highly resource-constrained, because any new actions that are planned must fit within a budget surplus within the resources allocated to the particular task being executed when the opportunity arose. In PANDORA, a structure (such as a chain segment) might be spotted during a traverse between mission components, giving rise to the opportunity to inspect that structure within available resource constraints.

In PANDORA, the command level planning is performed offline, by a deterministic plan generation system, POPF. Tactical planning is done online, when resource constraints and the actual locations of assets are known. Tactical plans are executed under adaptive dispatch and resource monitoring. Planning to exploit opportunities is also an online task, since opportunities are unforeseen, and the time required to build the plan is consumed from the available resource budget.

The opportunistic planning approach relies on resource-monitoring, so that unspent resources can be accumulated and reused for utility gain. The method by which this is done is discussed in the following sections. Figure 1.1 shows how the command and control

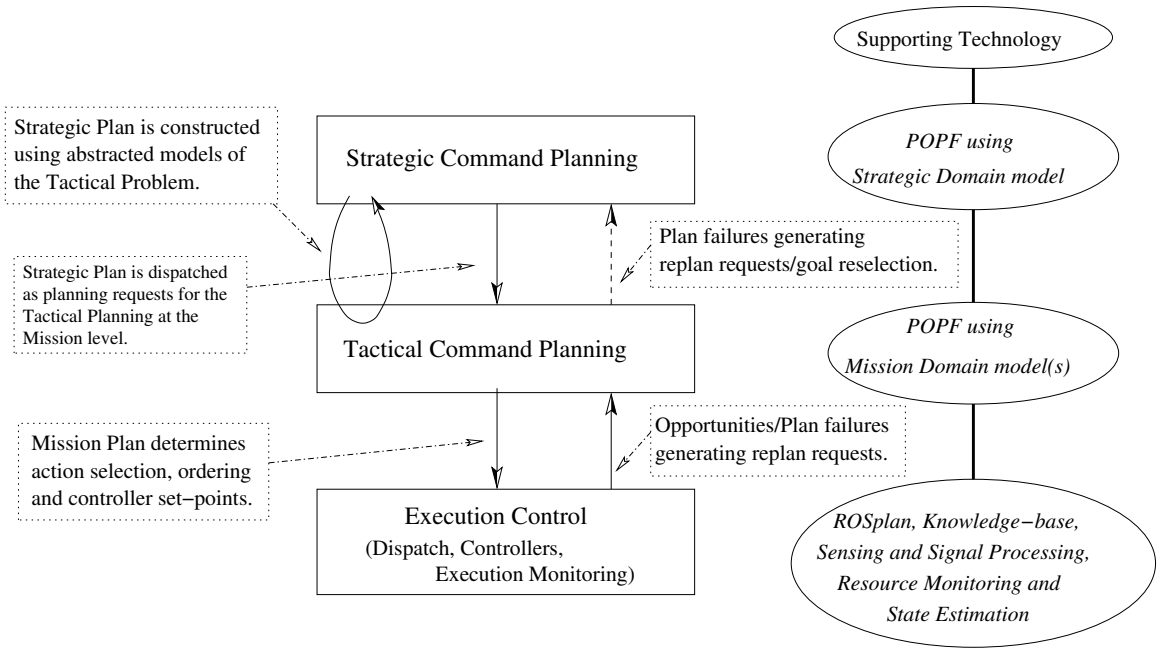


Figure 1.1: Command and Control planning in PANDORA. The Strategic Command Planning level uses abstractions of the Tactical Command models in order to obtain estimates of the resource requirements of the component missions. The Tactical level then plans each component mission within its resource envelope.

paradigm works in PANDORA.

1.1 The Opportunistic Planning Problem

The problem to be solved is a mixture of offline strategic planning and opportunistic management of high-impact, low probability events. The probability distribution over these HILPs is not known in advance.

The problem occurs in a temporal planning context, with windows of opportunity giving rise to *deadlines* that must be observed by the planner. Even though, in the PANDORA case, there is a single AUV executing actions in sequence, and therefore no concurrent activity, deadlines raise synchronisation issues which make online methods such as the online receding horizon approach [1, 13] impractical. The requirement to fit activities into time windows makes planning much harder than it would be without deadlines.

Opportunities can only be spotted and exploited during the execution of *preemptible* actions. In the PANDORA application, the only preemptible actions are the navigation actions (ie: opportunities only arise during traverses between waypoints).

The problem is defined by the following set of features:

- The system has a set of “hard” goals that is computationally hard to achieve (otherwise replanning is a good option).
- The system does not know the probability distribution over the opportunity space. It knows about the existence of certain types of opportunities that might arise, but not whether or where they will arise. The probability of any one of them is low, but exploring them might have very high utility.
- When an opportunity arises, it can only be addressed if the resource availability allows, because the top priority is to achieve the initial goal set.

When an opportunity arises, a new initial state will have to be constructed, containing new waypoints and object instances, which is a *monotonic extension* of the support set of the plan.

We now provide a formal model of the Opportunistic Planning Problem.

1.2 The Opportunistic Planning Model

Definition 1 *An* opportunity *is a tuple, $\langle T, L, Og, U \rangle$, where T is the name of a PDDL enumerated type, defined in the PDDL domain file, L is a 3-dimensional location, (x, y, z), Og is a goal, with one free variable, x of type T, and U is a utility value in $\mathbb{R}$. The goal $Og[x]$ is an opportunistic goal.*

Definition 2 *An* opportunity space*, $Opps$, is the set of all opportunities in a given planning domain.*

Definition 3 *A* probability density function (PDF) over an opportunity space *is a function:*

$$O : SO^3 \times N \times Opps \rightarrow [0, 1]$$

where N is the name of the action under execution, and the SO^3 term is the 3-dimensional coordinate and orientation at which the action is being executed.

Definition 3 models the likelihood that an opportunity that is present at a location will be spotted during the execution of an action. This model does not attempt to take account of the impact of time on observation: any activity executed at a single location over a period of time will have the same probability of spotting an opportunity irrespective of how long is spent executing the activity.

Definition 4 *An* abstracted probability density function *over an opportunity space is a function:*

$$O : SO^3 \times N \times Opps \rightarrow \{True, False\}$$

The abstraction reflects the fact that the real PDF is drawn from a family of PDFs in which the probabilities associated with the regions for which the abstraction returns True are higher (on average) than in other regions.

Definition 5 *An* opportunistic planning problem *is a tuple $\langle I, G, A, O_{Ab}, R \rangle$, where I is the initial state, G is a set of hard goals (they must all be achieved in any goal state), A is a set of grounded actions, containing both preemptible and non-preemptible actions, O_{Ab} is an abstraction of the unknown probability density function over* opportunities*, and R is a probability density function over* resource consumption*.*

Definition 6 *A* monotonic extension *of an initial state description, I, extends I with a new collection of objects, $O \cup \{o : T\}$, and facts F over O, and a new soft goal Og. If I is a tuple $\langle objects, init, H, S \rangle$, where H and S are the hard and soft goals, respectively, then the extended initial state, I', is the tuple $\langle objects \cup O, init \cup F, H, S \cup \{Og[x/o]\} \rangle$ and:*

$$\forall P \cdot (I \models P \rightarrow I' \models P)$$

Definition 7 *An* opportunistic plan fragment *is a plan constructed to achieve a grounded opportunistic goal.*

1.2.1 Relationship to other Probabilistic Models

We have considered two different models of opportunistic planning in our previous work. In the first case [6, 7], we assume that the opportunities and their locations are known in advance of starting the execution of the plan. Opportunistic plan fragments are computed offline, and executed online if their resource requirements are met. In the second case [14], we do not assume this, but instead we assume that we know the *types* of opportunities that can arise, and that all opportunities of the same type can be exploited by the same opportunistic plan fragment. These are precomputed and stored in a plan library. The relevant opportunistic plan fragment is then inserted into the plan whenever an opportunity of its type is identified, and resources allow.

The work we have developed in PANDORA is novel in three main respects. First, we do not assume that we know the distributions of the opportunities in advance, so they are genuine surprises to be discovered (or not) at execution time. Second, when an opportunity is spotted we plan *online* to exploit it, using a monotonically extended initial state description. Thus, opportunistic planning in the PANDORA framework fits into the theme of discovery and model refinement being explored throughout the project. Third, we have implemented our framework in the PANDORA planning approach, and performed an experimental comparison with a replanning strategy. As a final point, we have developed a formal model of opportunistic planning which provides a well-defined basis for further work.

There are several related works that tackle some aspects of our problem as stated, under the assumption that the probability distribution over the opportunity space is known. A well-explored method is *contingent planning*, in which alternative branches are built explicitly into the plan structure [12, 4]. Contingent planning is very expensive, so various methods have attempted to limit the number of contingent branches constructed. In particular, Coles [3] considers over-subscription planning with resource uncertainty. She builds tree-structured plans offline, where each branch is contingent on resource availability and leads to a different set of goals. An initial plan is constructed offline under conservative assumptions about resource use, and branches are then constructed, also offline, at points where the planned actions are assumed to have completed after consuming only the mean quantity of resources. In her approach, the configuration of the world and all goals, including opportunities and their locations, are known in the initial state. Burns *et al* [1] use an online receding horizon approach to consider anticipatory on-line planning in which plans take into account goals that are likely to arise, in order to be better prepared for achieving them efficiently.

When the probability distribution over the opportunity space is known, the problem can be modelled as a POMDP [8]. Whether or not to pursue an opportunity in a certain belief state amounts to whether the expected utility of pursuing the opportunity, in addition to achieving the hard goal set, all within the resource envelope available, exceeds the expected utility of completing the current plan under execution with lower resource pressure. The problem can be modelled but, even with recent work on improving efficiency [10, 11, 13], the decision-theoretic approach will not scale to the sizes of problems requiring to be solved in the practical application. But the offline decision-theoretic reasoning cannot be done at all, in the absence of knowledge about the probability distribution over the opportunity space. Knowledge of the abstracted PDF, O_{Ab}, is not sufficient for decision-theoretic reasoning because it does not differentiate adequately between the values of exploring different regions in the opportunity space.

All of these approaches, apart from replanning, assume that the opportunities PDF is known in advance. Replanning does not require any knowledge about probability distributions, either over the opportunity space or over the use of resources by actions. In a reactive online method, a replanning strategy responds to an opportunity by throwing away the plan under execution, and building a new, conservative, plan for the union of the hard goal set and the opportunity. It then executes this plan instead, whenever its available resources are

sufficient to achieve the new goal set.

Replanning is therefore a plausible approach to our problem. However, we predict that replanning will be unnecessarily expensive, because it will replan parts of the problem for which there is already a detailed, and resource-valid, plan structure in place. Planning itself consumes resources, so we expect it to be advantageous to minimise the amount of planning that is done online, so that resources can be spent on increasing the utility of the already planned mission.

Chapter 2

A New Approach for Opportunistic Planning

In this Chapter we present a new approach for opportunistic planning.

2.1 The Proposed Approach

Conservative planning is a method that seeks to exploit the classical planning framework, while simultaneously recognising the underlying, but unknown, stochastic behaviour of the execution environment. This means that well-researched methods in temporal-metric planning can be exploited, whilst ensuring a robust plan with a high degree of certainty of successful completion.

To plan conservatively, we consider the distributions of the action durations and plan at the 95th percentiles of these distributions. This means that we are highly conservative about the durations of the actions, which has the advantage of allowing enough time, in almost all cases, for the actions to complete. One of the advantages of this approach is that, if an action were to over-run then, depending on the variances in the remaining action distributions, it is likely that the time would be gained back from later actions executing in the plan. This would prevent failure of the plan as a whole, even though the individual action failed to complete in its planned duration.

Although conservative planning seems an inefficient way of allocating time to tasks, because the 95th percentile will only be reached on an infrequent basis, it provides a stable baseline which provides a robust confidence in the completion of the plan. It also provides a plan utility that is likely to be improved upon at execution time. After this baseline has been established, the executive may decide to use any time that it gains during execution to carry out extra tasks, such as pursuing opportunities, on top of the basic plan.

Rather than seeking to produce a plan with a high utility, and then modifying this plan to add robustness, opportunistic planning approaches the problem from the other direction, generating a robust plan, and then seeking to increase its utility.

Plans to exploit opportunities can be visualised as sub-plans (which might be long chains of actions) that branch off from the main plan trajectory, finally returning to the main plan at a point enabling its continued execution to result in the achievement of the hard goal set. These sub-plans are referred to as *opportunistic plan fragments*. Some steps from the main plan might be obviated by the opportunistic plan fragment, so that some reasoning is needed to return to the latest possible state on the main plan trajectory. Figure 2.1, part (a), shows an opportunistic plan fragment that has been inserted into the plan, while part (b) shows the structure of a contingent branching plan.

Execution of an opportunistic plan may be done via the use of an execution stack. When a decision is made to pursue an opportunity, the tail of the executing plan is pushed onto

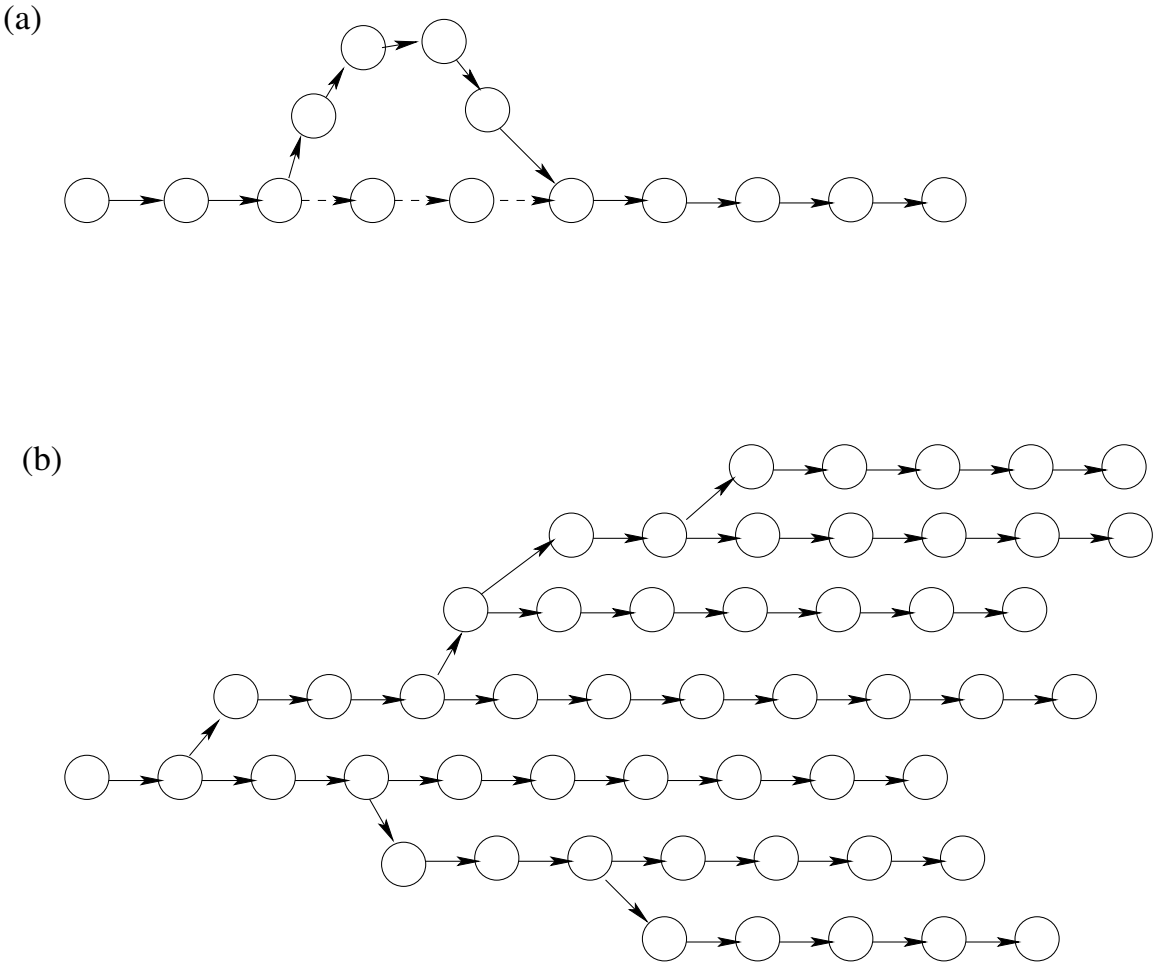


Figure 2.1: (a) The main plan with an opportunistic plan fragment attached. The fragment rejoins the plan suffix at the first necessary point for completion of the hard goal set. (b) The structure of a contingent plan. Each branch leads to a different goal set, depending on resource availability at the branch nodes.

the stack, and an opportunistic plan fragment is constructed. As long as this successfully completes within the planning time bound, and the resulting plan fragment can be executed within allocated resources, execution of the opportunistic plan fragment begins. When the opportunistic plan fragment has finished executing, the remainder of the main plan is popped off the stack, and its execution is then resumed. With this execution method, it is possible for opportunistic plan fragments under execution to be suspended and stacked, if a new opportunity is detected during its execution.

2.1.1 The Implementation

In our implementation of the above idea, we construct a strategic plan using conservative estimates of the resource requirements of actions. We use normal distributions over resource consumptions, in order to determine our conservative estimates. We do not model probability density functions over opportunity spaces. An abstract PDF, which captures that opportunities are more likely in open water than close to structures, is hard-wired into our setting. This is done by restricting the set of preemptible actions to contain only navigation actions. We can of course extend the power of the abstracted PDF by relaxing this

restriction.

The types of opportunities that can be identified in our PANDORA setting are `Chain` and `Pillar`. New objects of these types are created dynamically when opportunities are detected. The planner can perform certain actions on objects of these types, and instances of these actions can be created as soon as the object instances are known. No new action schemas are acquired, but new soft goals are created when opportunities are detected. These processes are described in Definition 6.

For operational reasons (Nessie and Girona-500 are differently equipped and cannot cooperate on any task in our setting), we assume a single AUV, equipped to do both inspection and intervention tasks. The planner uses the PDDL2.1 model defined in Appendix 3.1. This model contains the actions that can be performed by the AUV, and shows how temporal constraints, such as deadlines and action durations, are modelled.

Our implementation of the Opportunistic Planning method behaves as follows (see algorithm and figure 2.2):

- construct a sequential strategic plan to achieve the goal set (top level missions) within a conservative resource bound;
- start executing that plan under operational control, keeping track of unspent resources;
- branch off the plan to handle an opportunity within the unspent resource bound, storing the plan suffix (this is recursive);
- return to the plan suffix as soon as possible.

The execution of this algorithm, showing the management of the plan stack, is shown in Figure 2.3. When an opportunity is planned, the main plan suffix is pruned by removing all of the navigation actions at the front of the suffix. One non-general aspect of the approach so far is that we treat navigation actions as different from any other action, in that they are only needed to move the AUV to places where tasks can be done. They are never in the plan to achieve any top-level goal. They can therefore be removed from the suffix as long as we can reach the first waypoint at which an interesting action takes place in the suffix, on completion of an opportunity.

- input: time limit in seconds, missionID identifies the mission goals;
- line 1: now() is the current time. (at AUV MissionEndPoint) is included as a goal;
- line 3: replanRequested can be set true by external processes;
- line 12: We set by hand which actions can dispatch early (all except for turn_valve);
- line 16: AUVs are busy while still executing an action;
- line 18: An action's default timeout can be chosen per operator, either as duration*T for some T, or duration+T for some other T;
- line 22: opportunistic_plan_requested is a communication variable that is declared and initialised externall, and then set true by an external process;
- lines 26-29: These lines find the finishing location for the opportunistic plan, and remove the "goto" actions from the parent plan;
- line 32: If the opportunistic mission was not possible, then the "goto" actions are reinserted at the start.

2.2 Reactive Planning

We can contrast our approach with a replanning method, as shown in Algorithm 2.

Algorithm 1: runPlanningSystem

```
input  : timelimit : Int, missionID : Int, missionEndPoint : Waypoint
output : boolean
problem ← generateProblemFile(now(), missionID, missionEndPoint);
plan ← makePlan(problem);
replanRequested ← false;
freeTime ← 0;
if plan.length() > timelimit then
    return false;
else
    while plan.length() > 0 do
        currentAction ← plan.pop();
        dispatchTime ← currentAction.dispatchTime;
        if !canDispatchEarly(currentAction) and now() < dispatchTime and !replanRequested
        then
            wait();
        currentAUV ← currentAction.AUV;
        while AUV.isBusy() and !replanRequested do
            wait();
            if now() > currentAction.timeout then
                dispatch(cancelAction);
                replanRequested ← true;
            if opportunistic_plan_requested then
                opportunistic_plan_requested ← false;
                currentEndPoint ← currentLocation();
                prunedActions ← {};
                while plan.first() == "goto" do
                    currentEndPoint ← plan.first().destination;
                    prunedActions.push_back(plan.pop());
                plans.push_back(plan);
                if !runPlanningSystem(freeTime, opportunisticMissionID, currentEndPoint)
                then
                    plans.insert(prunedActions, 0);
                plans.pop(plan);
        if !replanRequested then
            dispatch(currentAction);
            freeTime ← now() - dispatchTime;
        else
            replanRequested ← false;
            problem ← generateProblemFile(now());
            plan ← makePlan(problem)
return true
```

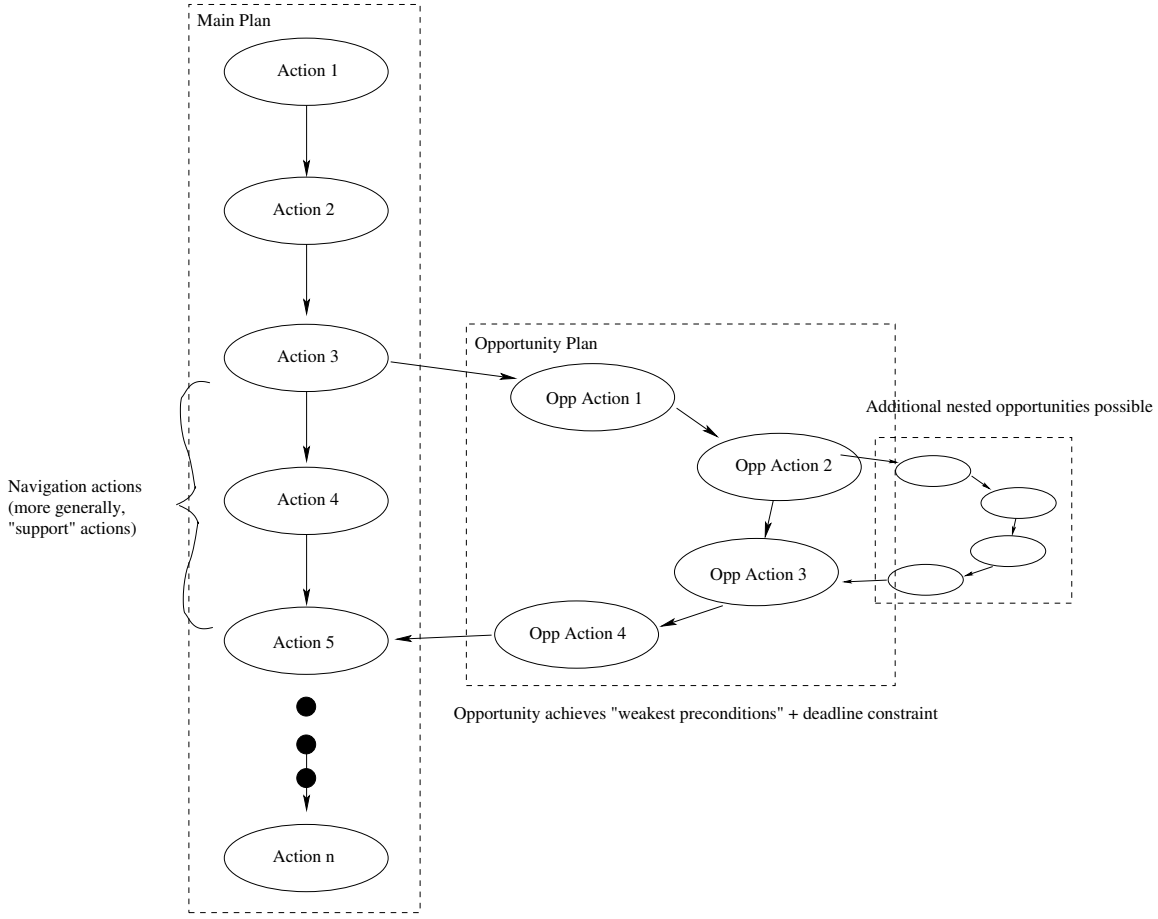


Figure 2.2: Plan execution with opportunity insertion.

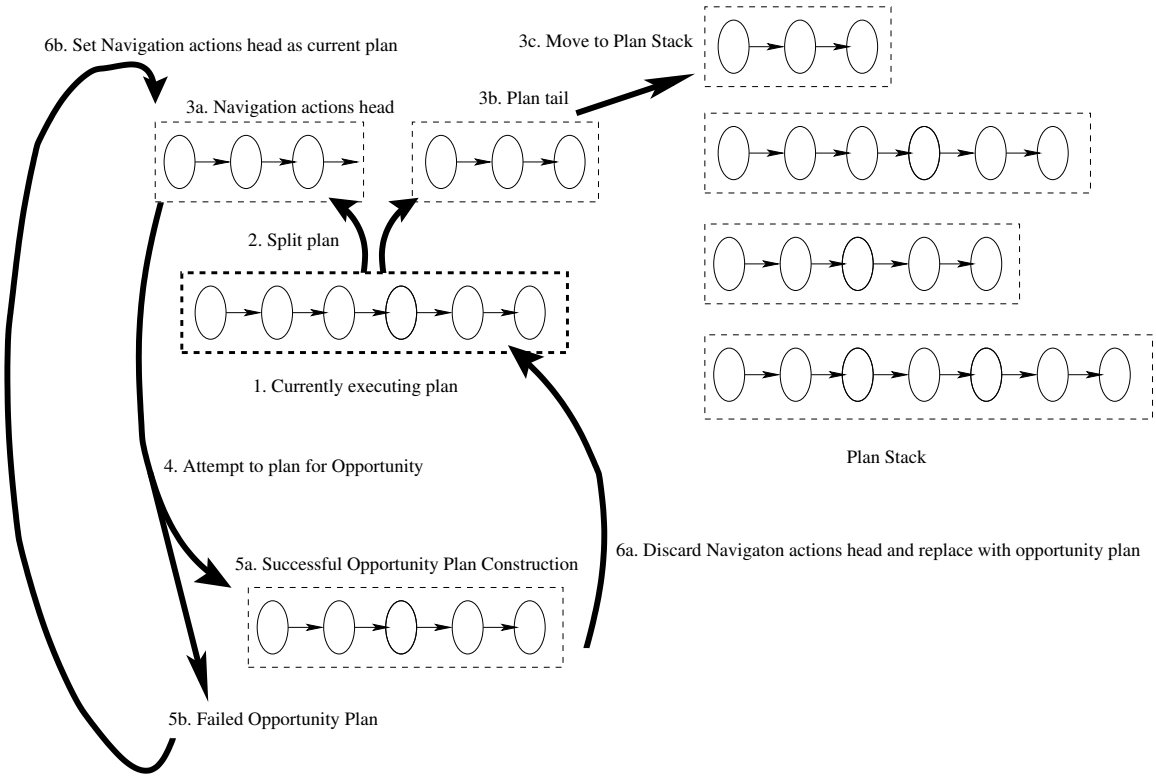


Figure 2.3: The execution of the algorithm, showing how plan suffixes are stacked. We start at 1 (the bold dashed rectangle), in the execution of the main plan. When an opportunity is detected, the goto actions are pruned and the plan suffix is stacked (2 and 3). Then we follow the process in numbered order. 5a and 6a follow from a successful opportunity construction, and 5b and 6b follow a failed attempt. The figure shows the plan stack (on the right).

Algorithm 2: runRePlanningSystem

```
input  : timelimit : Int, missionID : Int, missionEndPoint : Waypoint
output: boolean
problem ← generateProblemFile(now(), missionID, missionEndPoint);
plan ← makePlan(problem);
replanRequested ← false;
new_goals_created ← false;
freeTime ← 0;
if plan.length() > timelimit then
    return false;
else
    while plan.length() > 0 do
        currentAction ← plan.pop();
        dispatchTime ← currentAction.dispatchTime;
        if !canDispatchEarly(currentAction) and now() < dispatchTime
        and !replanRequested then
            wait();
        currentAUV ← currentAction.AUV;
        while AUV.isBusy() and !replanRequested do
            wait();
            if now() > currentAction.timeout then
                dispatch(cancelAction);
                replanRequested ← true;
            if new_goals_created then
                new_goals_created ← false;
                add_goals(missionID,get_new_goals());
        if !replanRequested then
            dispatch(currentAction);
            freeTime ← now() - dispatchTime;
        else
            replanRequested ← false;
            problem ← generateProblemFile(now());
            plan ← makePlan(problem)
return true
```

Chapter 3

Experiments and Results

Both the opportunistic strategy described here, and a basic replanning strategy, are reactive, in the sense that they do not attempt to plan for opportunities before they arise, but only once they are observed. In the case where no opportunity is observed during execution, the replanning strategy and the opportunistic planning strategy will both simply execute the main plan to achieve the hard goals, with no deviation (except in response to plan failure, which we ignore here). Therefore, the differences lie only at the point where an opportunity is discovered. In the replanning case, we construct a new initial state and replan for the entire goal set. In the opportunistic planning case, we plan only for the opportunity, together with a goal to return to the start of the remaining suffix of the rest of the plan. Both approaches begin by constructing the monotonically extended initial and goal states. The opportunistic approach benefits from what is usually a simpler planning problem in exchange for the possibility of finding a better plan that can exploit the remaining resources to achieve the opportunity and original goals more efficiently. Our experiments consider the situation at a point in which an opportunity has been discovered and the two strategies are attempting to decide whether to pursue it or not.

We perform the comparison by setting up a main mission, with hard goals, and an opportunity. The main mission is taken to be either a valve-turning mission, and the opportunity mission is an inspection or a chain-following mission. We report here only the results for inspection opportunities: results for the chain-following task using the replanning mission are not directly comparable with the opportunistic strategy, because of the structure of the chain-following task, which can be assigned variable duration according to available resources. These missions comprise three of the core PANDORA missions. In the valve-turning mission, the AUV is required to approach and set two valves within a deadline. The effect of setting a deadline is to bound the resource available for exploiting opportunities. The inspection mission is not time-limited. When it arises as an opportunity, a plan to exploit it must fit within the available resource envelope. Inspection missions are of several sizes, ranging between 2 and 32 inspection points. A chain following mission consists of navigating to a link that has been spotted, and then following the chain for a period of time.

The mission elements, both main and opportunity, are within an area 50m by 50m and set at least 5m apart. They are positioned successively, with uniform probability over the available area. The deadlines for valves are set to different values, making the planning problems harder as the deadlines are tightened. The opportunistic planning strategy requires the opportunity to be exploited within the free resource window, *before* the completion of other mission components. The replanning strategy does not require this, but both strategies require the overall plan to be completed by the mission deadline.

In Table 3.1 we report our results for a collection of randomly generated problem instances. The opportunistic planner is given 10 seconds to solve the problem. In general, the window of opportunity is short, partly because it is most often the case that we will discover

an opportunity while navigating, in which case we do not want to stop the vehicle unless we decide to pursue the opportunity, and partly because the energy and computational resources on board the AUV is limited. It is also important that the time taken evaluating an opportunity should not be significant compared with execution time of actions, otherwise we endanger the main mission itself by wasting resources on multiple opportunity evaluations. This latter problem is particularly an issue if the signal processing that leads to recognition of an opportunity is unable to solve the registration problem that would enable it to determine that multiple sightings of the same object are actually not distinct opportunities.

The replanning strategy was allowed 30 minutes of CPU time to generate a best possible plan. We report the best plan found in that time, with the time it took to find that plan (POPF uses an anytime strategy of plan improvement, reporting plans as they are found).

As the results indicate, the problem with replanning is that planning the combined mission is typically too difficult to complete within the 10 second bound. This problem would, of course, be far more acute if the main mission were more complex (as will usually be the case in practice). The difficulty in planning the combined mission often leads to the replanner producing plans that are actually much worse than the cost of the opportunistic plan fragment added to the cost of the main mission plan suffix (shown as the duration of the Opportunistic Complete Plan), despite the fact that the latter cost represents an upper bound on the cost of the best mission for the complete set of goals.

The bolded results are the cases in which the combined mission is solvable with a higher quality solution within the 30 minute bound. In four of these cases, the replanning strategy would outperform the opportunistic strategy, but in the bold and italicised case, the plan takes so long to find that the combined planning and execution time exceeds the time available for the complete plan. Indeed, in almost all cases, the complete plan is so much longer than the opportunistic plan that it would not be possible to complete within the duration of the intended mission time for the whole problem. Part of the difficulty for the replanning strategy is that the existence of deadlines leads to the planner pushing activity later along the time line than is appropriate and it fails to search the parts of the search space in which the short plans exist. This is, of course, a weakness of the planner, and we will, in future work, explore improved temporal planning strategies. Unfortunately, the problems we are exploring lie outside the current capabilities of any planners other than POPF.

In one of these test cases the opportunistic planner failed to find a plan within 10 seconds, so the plan reverts to the main mission plan. In this case, the replanning strategy takes 3 minutes to find a plan that is far too long to be used in place of the main mission plan, so this represents a waste of the time spent in this attempt.

These results show very clearly that the cost of a complete replan is much higher than planning for an opportunity alone. Even though planning for the combined mission should offer, in principle, a chance to find a better quality solution than the one found by simply linking the opportunity plan to the front of the existing plan, the reality is that it is very hard to achieve this. A more capable planning strategy might be more successful in finding better plans, but the time taken to do so would certainly be far greater than the time required to find the opportunistic plan. Each such plan construction attempt spends the very resource that is required to exploit the opportunity itself, so it is an impractical approach to repeatedly evaluate opportunities by using a full replanning approach.

3.1 Future Work

Our current approach to opportunistic planning demonstrates improvements over a replanning strategy, but has some limitations. In particular: we do not evaluate the expected gain, in terms of accumulated resource, of reducing our confidence in achievement of the hard goal set. For example if, at some point p, into the execution of a plan, we are willing to

Mission		Opp plan	Full replan	Plan duration		
Main	Opp	time	time	Opp Mission	Complete Opp Plan	Replanned plan
V2_400	I16	0.36	38.18	851.384	1265.032	2437.496
V2_500	I16	5.54	7.46	1541.168	2076.155	2596.156
V2_600	I16	5.34	7.28	1541.168	2117.136	2269.701
V2_700	I16	5.32	9.56	1541.168	2117.136	2283.134
V2_800	I16	5.38	6.24	1541.168	2117.136	**2048.833**
V2_900	I16	5.4	9.16	1541.168	2117.136	**1900.069**
V2_1000	I16	0.38	21.42	851.384	1265.032	2615.245
V2_1100	I16	0.34	7.28	888.554	1302.202	2048.833
V2_1200	I16	2.4	11.9	1440.568	1854.216	2511.960
V2_1300	I16	0.36	6.34	851.384	1265.032	2772.985
V2_1400	I16	0.42	6.28	851.384	1265.032	2772.985
V2_1500	I16	0.34	7.82	851.384	1265.032	2946.391
V2_1600	I16	0.38	14.54	851.384	1265.032	2175.901
V2_1700	I16	0.4	15.6	851.384	1265.032	2897.665
V2_1800	I16	0.42	6.24	851.384	1265.032	2772.985
V2_1900	I16	0.38	6.44	851.384	1265.032	2772.985
V2_2000	I16	0.36	2.62	851.384	1265.032	2490.490
V2_400	I32	5.08	148.17	2233.961	2564.254	3531.784
V2_500	I32	2.2	165.62	1768.98	2129.213	5332.514
V2_600	I32	3.7	78.19	1777.177	2137.41	3623.974
V2_700	I32	4.08	272.84	1815.849	2176.082	4877.45
V2_1000	I32	4.66	104.04	2686.638	3093.992	4263.605
V2_2000	I32	4.32	100.16	2457.922	2865.276	3778.601
V2_2500	I32	4.67	68.78	2457.922	2865.276	4212.37
V2_3000	I32	4.36	132.32	2469.124	2876.478	3948.493
V2_3500	I32	4.21	119.14	1861.244	2191.537	4925.07
V2_5000	I32	5.16	81.04	1997.34	2327.633	5460.531
V2_1000	I2	0.02	3.36	141.138	678.167	**580.58**
V2_1000	I6	0.06	182.02	374.415	911.444	***774.337***
V2_1000	I8	0.06	4.82	504.346	863.481	977.001
V2_1000	I10	0.1	135.62	582.795	1017.846	1383.791
V2_2000	I10	0.1	7.44	700.198	1294.007	1585.303
V2_2000	I12	0.14	3.38	772.926	1545.852	**1414.551**
V2_2000	I14	0.14	3.70	675.458	1141.406	2040.448
V2_2000	I16	0.14	2.60	676.458	1142.406	2490.490
V2_2000	I18	0.18	44.68	878.395	1470.264	3116.591
V2_2000	I20	0.44	15.18	1231.22	1767.021	3358.226
V2_2000	I22	1.08	17.42	1752.10	2152.423	4114.774
V2_2000	I24	0.98	34.48	1643.92	2017.172	2914.554
V2_2000	I26	2.6	289.40	2141.87	2485.068	6029.480
V2_2000	I28	3.38	265.72	3088.31	3667.984	5987.822
V2_2000	I30	-	179.76	-	398.45	4187.185
V2_2000	I32	4.14	218.74	2689.84	3057.283	4475.005
V2_2000	I34	5.24	89.32	3125.49	3621.695	4615.062

Table 3.1: Table of experimental results. Planning time and plan durations are measured in seconds.

reduce our confidence in successful execution of the plan suffix to the 94th percentile, how much resource could we save for spending on an opportunity spotted at p?

As a plan containing multiple actions is executed, the use of the 95th percentile as an estimate for the nominal execution time of each action leads to an accumulating expected error. So, if k actions all with mean execution time m and standard deviation s are executed in sequence, the nominalised times will yield a total time for execution of $k(m + 1.65s)$. The time actually required to achieve the 95th percentile for the combined sequence of actions is only $km + 1.65\sqrt{k})$, showing that the estimate based on individual 95th percentiles yields a $1.65s(k - \sqrt{k})$ over-estimate of the time required for the 95th percentile. As a practical example, suppose that navigating the traverse between two waypoints on the installation has a mean time of 11507 seconds, and a standard deviation of 925 seconds. If 5 successive traverses between waypoints are to be executed, the use of the nominal time estimates will yield an estimated duration of 65.165 seconds. The 95th percentile for the estimated duration of the combined sequence is 60.948 seconds, so using the 95th percentile for nominalisation will lead to an expected overestimate for the execution time of just under four and a quarter seconds 7% of the 95th percentile time for the execution of the complete sequence. As an alternative to allowing the expected accumulation of resources following the execution of a sequence of nominalised actions it would be possible to adjust the nominalised values to account for the length of the sequence. So, for k actions each with identical mean and standard deviation, the nominalised durations can be reduced to $m + \frac{1.65s}{\sqrt{k}}$. More generally, where several actions are sequenced to achieve a goal it is possible to discount the nominalised time to allow for the expected accumulated benefits of using the 95th percentile as the nominal durations of the individual components.

In our future work we intend to experiment with trading off confidence against utility, by doing this reasoning *online* at the point at which we have evaluated an opportunity. The actions in the plan suffix are not changed, but the confidence in completing it successfully is traded for the benefits of the opportunity. For a very high value opportunity it might even be worth, in order to free up more resource, requesting the sacrifice of a component mission from the command level planner (see the dotted arrow in Figure 1.1.

Acknowledgements

The authors would like to thank colleagues in the PANDORA consortium. Pandora is supported by European Commission grant number FP7-ICT-2011-7 Ref. 288273.

Appendix: PDDL2.1 Model

This PDDL2.1 fragment shows how the PANDORA mission domain is modelled to the planner. The actions are schemas, which can be instantiated in different ways depending on the objects defined for the problem. The actions are *durative*, meaning that they take time to achieve their effects. The action durations are determined experimentally. The durations of the two navigation actions, `do_hover_controlled` and `do_hover_fast`, correspond to the time it takes to travel between pairs of waypoints in the model. The duration of the `turn_valve` action reflects the time taken to complete an execution of the learned valve-turning action.

```
(define (domain pandora)

(:requirements :strips :typing :fluents :disjunctive-preconditions
               :durative-actions :timed-initial-literals
               :duration-inequalities)

(:types
  waypoint
  inspectionpoint
  pillar
  panel
  valve
  chain
  vehicle)

(:predicates

 (waypoint_not_occupied ?wp - waypoint)
 (connected ?wp1 ?wp2 - waypoint)

 (at ?v - vehicle ?wp - waypoint)
 (near ?v - vehicle ?wp - waypoint)
 (not_illuminating ?v - vehicle)

 (cansee ?v - vehicle ?wp - waypoint ?ip - inspectionpoint)

 (cansee_pillar ?v - vehicle ?wp - waypoint ?p - pillar)
 (observed_pillar ?p - pillar)
 (pillar_illuminated ?p - pillar)

 (canexamine ?v - vehicle ?wp - waypoint ?p - panel)
```

```
 (canreach ?v - vehicle ?wp - waypoint ?p - panel)
 (examined ?p - panel)
 (on ?a - valve ?p - panel)
 (valve_blocked ?a - valve)
 (valve_free ?a - valve)

 (chainat ?c - chain ?wp - waypoint)
 (chain_examined ?c - chain)
)

(:functions

  (arm_calibration ?auv - vehicle)

  (observed ?ip - inspectionpoint)
  (obs ?ip - inspectionpoint ?wp - waypoint)

  (distance ?wp1 ?wp2 - waypoint)

  (valve_goal ?va - valve)
  (valve_state ?va - valve)
  (valve_goal_completed ?va - valve)

  (speed ?v - vehicle)
  (fast_speed ?v - vehicle)
  (observe-time ?v - vehicle)
  (adjust-time ?v - vehicle)
  (inspection-time ?v - vehicle)
  (illuminate-time ?v - vehicle)
  (valveturn-time ?v - vehicle)
  (recalibrate-time ?v - vehicle)
  (chain-time ?v - vehicle)

)

(:durative-action do_hover_controlled
 :parameters (?v - vehicle ?from ?to - waypoint)
 :duration ( = ?duration (* (distance ?from ?to) (speed ?v)))
 :condition (and
    (at start (at ?v ?from))
    (at start (connected ?from ?to)))
 :effect (and
    (at start (not (at ?v ?from)))
    (at end (at ?v ?to)))
)

(:durative-action do_hover_fast
 :parameters (?v - vehicle ?from ?to - waypoint)
 :duration ( = ?duration (* (distance ?from ?to) (fast_speed ?v)))
 :condition (and
    (at start (at ?v ?from))
    (at start (connected ?from ?to)))
 :effect (and
    (at start (not (at ?v ?from)))
```

```
    (at end (near ?v ?to)))
)

(:durative-action correct_position
 :parameters (?v - vehicle ?target - waypoint)
 :duration ( = ?duration (adjust-time ?v))
 :condition (and
    (at start (near ?v ?target)))
 :effect (and
    (at start (not (near ?v ?target)))
    (at end (at ?v ?target)))
)

(:durative-action observe_inspection_point
 :parameters (?v - vehicle ?wp - waypoint ?ip - inspectionpoint)
 :duration ( = ?duration (inspection-time ?v))
 :condition (and
    (at start (at ?v ?wp))
    (at start (cansee ?v ?wp ?ip)))
 :effect (and
    (at start (not (cansee ?v ?wp ?ip)))
    (at start (not (at ?v ?wp)))
    (at end (increase (observed ?ip) (obs ?ip ?wp)))
    (at end (near ?v ?wp)))
)

(:durative-action illuminate_pillar
 :parameters (?v - vehicle ?wp - waypoint ?p - pillar)
 :duration ( = ?duration (illuminate-time ?v))
 :condition (and
    (at start (at ?v ?wp))
    (at start (cansee_pillar ?v ?wp ?p)))
 :effect (and
    (at start (pillar_illuminated ?p))
    (at start (not (not_illuminating ?v)))
    (at start (not (at ?v ?wp)))
    (at end (not (pillar_illuminated ?p)))
    (at end (not_illuminating ?v))
    (at end (near ?v ?wp)))
)

(:durative-action observe_pillar
 :parameters (?v - vehicle ?wp - waypoint ?p - pillar)
 :duration ( = ?duration (observe-time ?v))
 :condition (and
    (at start (at ?v ?wp))
    (at start (cansee_pillar ?v ?wp ?p))
    (at start (not_illuminating ?v))
    (over all (pillar_illuminated ?p)))
 :effect (and
    (at start (not (cansee_pillar ?v ?wp ?p)))
    (at start (not (at ?v ?wp)))
    (at end (observed_pillar ?p))
    (at end (near ?v ?wp)))
```

```
)

(:durative-action examine_panel
 :parameters (?v - vehicle ?wp - waypoint ?p - panel)
 :duration ( = ?duration (observe-time?v))
 :condition (and
    (at start (at ?v ?wp))
    (at start (canexamine ?v ?wp ?p)))
 :effect (and
    (at start (not (canexamine ?v ?wp ?p)))
    (at end (examined ?p)))
)

(:durative-action turn_valve
 :parameters (?v - vehicle ?wp - waypoint ?p - panel ?a - valve)
 :duration ( = ?duration (valveturn-time ?v))
 :condition (and
    (at start (at ?v ?wp))
;     (at start (examined ?p))
    (at start (canreach ?v ?wp ?p))
    (at start (on ?a ?p))
    (at start (<= (arm_calibration ?v) 1))
    (at start (valve_free ?a)))
 :effect (and
    (at start (not (valve_free ?a)))
    (at end (assign (valve_state ?a) (valve_goal ?a)))
    (at start (not (at ?v ?wp)))
    (at end (near ?v ?wp))
    (at end (increase (valve_goal_completed ?a) 1))
    (at end (increase (arm_calibration ?v) 1))
    (at end (valve_blocked ?a)))
)


(:durative-action follow_chain
 :parameters (?v - vehicle ?c - chain ?from - waypoint)
 :duration ( = ?duration (chain-time ?v))
 :condition (and
    (at start (at ?v ?from))
    (at start (chainat ?c ?from)))
 :effect (and
    (at start (not (at ?v ?from)))
    (at end (chain_examined ?c))
    (at end (near ?v ?from)))
)

)
```

Bibliography

[1] Ethan Burns, J. Benton, Wheeler Ruml, Sunwook Yoon, and Minh B. Do. Anticipatory Online Planning. In *Proceedings of 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

[2] A. Coles, A. Coles, M. Fox, and D. Long. Forward-chaining Partial-order Planning. In *ICAPS 2010 - Proceedings of the 20th International Conference on Automated Planning and Scheduling*, pages 42–49, 2010.

[3] Amanda Coles. Opportunistic Branched Plans to Maximise Utility in the Presence of Resource Uncertainty. In *Proceedings of European Conf. on AI (ECAI)*, 2012.

[4] Mark Drummond, John Bresina, and Keith Swanson. Just-in-Case Scheduling. In *Proceedings of 12th National Conference on Artificial Intelligence (AAAI)*, pages 1098–1104, 1994.

[5] M. Fox and D. Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[6] Maria Fox and Derek Long. Single-trajectory opportunistic planning under uncertainty. In *UK Planning SIG, Delft*, 2002.

[7] Jonathan Gough, Maria Fox, and Derek Long. Plan execution under resource consumption uncertainty. In *Proceedings of the ICAPS Workshop on Connecting Planning and Execution*, pages 24–29, 2004.

[8] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Articial Intelligence*, 101:99–134, 1995.

[9] Bernice Lee, Felix Preston, and Gemma Green. Preparing for High-Impact, Low-Probability Events: Lessons from Eyjafjallajökull, 2012.

[10] S. Ong, S. Png, D. Hsu, and W. Lee. POMDPs for robotic tasks with mixed observability. *Robotics: Science and Systems*, 2009.

[11] J. Pineau, G. Gordon, and S. Thrun. Anytime point-based approximations for large POMDPs. *Journal of Articial Intelligence Research*, 27:335–380, 2006.

[12] Louise Pryor and Gregg Collins. Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research*, 4:287–339, 1996.

[13] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for pomdps. *Journal of Articial Intelligence Research*, 32:663–704, 2008.

[14] Mark Woods, Andy Shaw, Dave Barnes, Dave Price, Derek Long, and Derek Pullan. Autonomous science for an ExoMars Rover-like mission. *J. Field Robotics*, 26(4):358–390, 2009.