Leaving Choices Open in Planner/Planner Integration*

Sylvain Joyeux and **Frank Kirchner**
DFKI Bremen, Robotics Innovation Center
Robert-Hooke-Strae 5
D-28359 Bremen

Abstract

In this paper, we analyze the shortcoming of some of the hierarchical planner/planner integration schemes in use nowadays. We will show that, in these integration methodologies, one planner has usually to take decisions that are not limited to its "area of expertise" (path planning, low-level planning, high-level goal scheduling, ...). It can therefore make ill-informed decisions and closes open choices that could have been used for planning and scheduling. We propose a different view on planner/planner integration, where one planner does not provide a single solution but instead filters its plan space, returning a set of near-optimal solutions. In other words, it returns the solutions that are *sensible* according to its model. The actual choices being then made either by other planners that use this information, or during execution when missing relevant information is made available to the system. Changing the point of view in this way would for instance allow a robot to proactively improve its knowledge to make better informed decisions.

1. Introduction: Decision-making in Robotic Systems

What is decision making in robotic architectures is a difficult question, and this section will definitely not give a definitive answer. It will instead try to give, through examples, a feeling for the following three categories that can be found in a robotic system:

- components definitely *not* taking decisions.
- components definitely taking decisions.
- a "grey area" where the answer is not so clear

No decision making Low-level behaviours like "go forward" or "turn left" are obviously *not* decision-making components. Another example is a module to control motors, or a pilot module for an UAV. They do not *choose* anything: they take a very short-term command (for instance the desired path the robot should follow) and tries to make the system stick to that command regardless of external influence (in case of side wind for an UAV for instance).

*Work done in the Intelligent Mobility project, funded by the German DLR, with funds from the Ministry of Science (BMBF).

Decision-making components Task planning deals with the selection of the future actions of the robot, given its goals, its current state, a model of its environment and a model of how it can change its own state and its environment. Action planning is definitely in the domain of decision making: it chooses one combination of actions that will allow the system to reach its stated goal(s) *among all the possible combinations of actions*.

Grey area In there lies *path planning*, or – more generally – all processes which involve choosing the place the robot should move. This might be, for instance, a *perception planner* which tells where perception should be done. These components get a goal and deal with choosing the right path for the robot to reach it. One could argue that given the environment model, the planner's cost function and the geometrical model of the robot, there is only one optimal path. However, when taking into account the actual path execution, one can easily that this optimal path will in the end prove to be only one among a set of *qualitatively equivalent* solutions. Thus, from the robot mission point of view, most path planning algorithms choose a single path in all the possible equivalent ones. They make a decision, and this decision impacts the whole system.

This actually could be seen as a definition of decision-making. Any component which *chooses* one of the possible, sensible courses of actions makes a decision. As such, they are all *planners*: they do a projection of the robot into its possible futures and choose one (or many) among them. As we will see in the next section, it is important to understand the interaction between the different decision-making components – and to allow them to have rich interactions – as these interactions are critical for the overall robot performance.

The next section will focus on issues that we identified in the integration between planning components. Then, we will develop the example of path- and task- planning integration through the presentation of a qualitative path planning algorithm we developped. Finally, the generalization of this approach to the integration of task planners will be discusses, and we will conclude on the general usefulness of our approach.

2. Problem Statement

In most systems, path planners are integrated using what one could call the "MoveTo(goal)" paradigm. As we outlined in the previous section, they are asked to reach a place, and choose one solution that would allow the robot to reach that place. For that scheme to work flawlessly, the path planner would have to concern itself with problems that go out of the "robot motion" domain, as for instance possible robot-robot interactions or scheduling issues: it has to consider a whole-system point of view and not only a navigation point of view.

For instance, a good solution for the classical rover-AUV navigation scenario – where the rover has a goal to reach and the AUV can provide some mapping information to it – is to have the rover take into account the AUV's task scheduling in its navigation decision, i.e. decide to take a route over another based on *if* and *when* the AUV will be able to provide him some data about the possible routes. Additionally, if it is presented with the relevant information, the task planner may be able to schedule information gathering tasks that would help the system in its decision. However, for both solutions to be easily integrated, the task planner and the multi-robot cooperation subsystem would have to know what options are actually open to them.

This problem also appears between action planners: in architectures where action planning is split into hierarchies – as for instance in (Alami et al. 1998; Alami and Botelho 2001) – the lower layers have only a partial knowledge of the plans of the upper layers: their knowledge is limited to the actions that should be executed right now. In these schemes, there is not the possibility to choose, in the lower layers, the optimal actions *given the future plan of the upper layer*. The lower layers can therefore over-constrain the upper layer.

To summarize our point here, decision are not always made in the component that is the most well-informed to make it. Note that this issue remains in systems where the planning and execution layers are tightly integrated through the building of a single plan or the sharing of partial plans as in CLARAty (Volpe et al. 2000; 2001; Nesnas et al. 2005), IDEA (Muscettola et al. 2002; Finzi, Ingrand, and Muscettola 2004) and the Concurrent Reactive Plans (Beetz 2000). It remains, because for the planning problem to be tractable, one *must* split it into smaller sub-problems and, until now, that involves using some domain-specific components, planning algorithms or cost functions, to make decisions while taking into account parameters unrelated to their focus. These components are then making choices that should have been done by a more informed or better suited layer, and also leads to a complexification of the cost functions, which become a delicate balance between all the trade-offs that the system faces.

2.1 The Example of Path Planning Integration

First, most path planners search for an optimal solution in the geometrical space (for instance in traversability maps) while, given the scale they are operating at, searching optimality is actually meaningless. Indeed, the uncertainty on the travel time or energy consumption is *far greater* than the difference between the optimal path and the next-to-optimal.

Second, as we already detailed, limiting the output of path planners to the optimal path hinders the ability to use task planning for information gathering, multi-robot cooperation, and complexifies the design of cost functions.

We therefore advocate to start thinking as path planner as a *filter*: it takes the whole geometrical (or configuration) space and finds out the *set of solutions* that are qualitatively equivalent from the point of view of the navigation's subsystem. Indeed, these solutions will be, in practice, equivalent at execution time and therefore a choice cannot be made by pure "travel-oriented" concerns (time, energy).

So, the path planner would output a set of solutions, leaving the actual choice to another component (task planner, plan manager). The advantage is then that one can take into account concerns that would be difficult to put in the path planner's cost function. For instance:

- in multi-robot contexts, *opportunistic* behaviors during path execution: for instance, taking photos of a place that another agent requires without impairing the robot's main mission.
- generation of contingency plans, where the task planner takes into account both a nominal movement and the ability to use alternate paths if the nominal one is blocked.
- the proper handling of unknown regions. In outdoor environments, the robot badly knows the environment itself. It must therefore decide whether it should explore an unknown (or badly known) region or if it should take a known path that is potentially longer. This decision is currently implicitly made by the system's path planner through the use of *a priori* values for the unknown parts of the map. In other words, the robot designer assumes the unknown to be of a certain type. An integrative approach could instead use the task planning system that has access to both the mission profile (is there enough time and energy for exploration?), and the consequences of a failure (what would be the cost of a failure?) to make that decision.

Finally, the *stability* of a path plan is of great importance for their integration in task plans. Optimal approaches, like D*-based approaches, become unstable around "crossroads" (where two completely different paths of almost equal costs meet each other). Indeed, little changes in the robot's environment model, or position estimate, can make the path costs change in a way that make the algorithm switch between them.

Because of this instability, it is hard to use the generated path in the task plan, as the latter can change dramatically at any time. For instance, in multi-robot contexts, there is no way to assume how a robot is committed to its currently chosen path. The authors have already observed the consequences of such instability in a bi-robot setup where the UAV uses the rover's path to generate the regions to map (Joyeux, Lacroix, and Alami 2007), which led to the UAV having to constantly move to completely different places. Again, by knowing the set of solutions from which the rover's execution engine will pick, the multi-robot cooperation component would be able to mitigate that problem.

2.2 Proposed Integration Scheme

What we basically advocate in this paper to change the way planners are integrated with each other. Through the path planner example, we will see that a hierarchical planner/planner integration can be achieved by having the planners acting as *filters*. One planner no more outputs one single near-optimal solution, but a set of solutions among those that are sensible (which often would translate as "near-optimal") given the level of detail of the considered planner. For instance, path planning should only have to reason on navigation optimization criteria (time, energy), allowing higher level tools to inject other criteria (cooperation possibilities, contingencies).

The matter of integrating choices in plan representations is not something new in itself. One can see conditional planning and MDP as allowing the representation of open choices by defining "choice tasks" which have multiple outcomes. The issue, in our opinion, is not to represent choices but to represent the information needed for scheduling. Indeed, as we mentioned, one unique feature of representing the choices in plans is that one can then represent the actions that are required to make informed choices. There would be, inherently, the need to have the planner stop seeing the plan as a static problem (static forward model) but really as a dynamic problem where tasks can influence decisions. The GPGP/TAEMS (Lesser et al. 2004) is an example of such a plan model.

The following section will further develop the example of path- and task- planning integration, by presenting our actual implementation of the proposed scheme. Then, we will discuss the possible generalization of such a scheme to the realm of task planning. Finally, we will conclude with the limitations and possibilities of our approach.

3. Choices in Plans

In this section, we will see how we represent and generate these choice plans in the case of the path planning problem, and how those plans have been integrated into our plan manager (Joyeux et al. 2009). The next section will then discuss the generalization to the integration of task planners, as well as open questions that exist in this approach.

3.1 Corridor Plans

The basic idea behind our representation is that planning for navigation should be *qualitatively* optimal: the goal is not to try following the optimal trajectory, but to represent and structure the regions of space where the robot should progress for the overall movement to be near-optimal. The resulting plan will therefore be both a geometrical representation of this region and the necessary symbolic information to direct the robot in it.

At the geometrical level, this region is segmented in "corridors". A corridor is a virtual tunnel, which has a *median line* and two boundary curves. From a navigation point of view, if the robot enters a given corridor from one end, it will traverse it and exit it at the other end (Fig. 1). I.e. the task plan will see one corridor as an atomic action.

Then, the corridors are structured into a topological plan, which represents how the corridors can be chained during execution. It links the corridor end regions with two types of edges:

- edges that represent the possible direction of travel inside one corridor (intra-corridor edge).
- connections between two end regions of two different corridors (inter-corridor edges).

In this representation, a well-formed *path*, that is a corridor sequence, is an alternance of edges of the two types. In other words, once an inter-corridor edge is taken (i.e. move from one corridor to another), the robot must take an intra-corridor edge (i.e. cross the corridor), and vice-versa. This guarantees that the robot won't go back into the corridor it just crossed.

Corridors can therefore be bidirectional if it exists two paths that traverse it in the two opposite directions. Or they are directed if all paths that traverse it do it in the same direction. In the first case, there are two intra-corridor edges (one for each direction of travel), and in the second case only one.

While similar to topological maps (see (Thrun 1998) for both an interesting method around topological maps and the related work in that field), it is not a representation of a map, but of a plan: it is a compact representation of the set of corridor sequences (plan executions) that lead to the robot's goal in a near-optimal way. Our contribution here is that we are able to generate such a plan in an unstructured outdoor environments, using the cost function to structure the space. The downside of it being that, like all plans, the topological representation will probably change if the goal changes while the environment does not.

3.2 Generation of Corridor Plans

For our plan generation algorithm, we assume that the environment is represented as a navigation function sampled on a grid map. The navigation function gives the optimal cost to the goal for each point in the map and can for instance be generated by D* or one of its offspring (we're using D*). The first phase of our algorithm transforms the navigation function into a total cost function: for each point, the algorithm computes the added cost of first going to that point and only then going from that point to the goal. Then, the algorithm marks the "navigation region", which includes all points whose *total* cost is within a given margin of the optimal cost. That "cost margin" can be set in different ways, as for instance (i) a fixed value which represents the allowed margin on the overall cost, (ii) a ratio of the optimal cost to goal or (iii) based on the uncertainty of the optimal path (if uncertainty informations is available).

The navigation region is then structured by using a Voronoi skeleton extraction algorithm. Voronoi diagram extraction is a widely studied subject, and we therefore do not describe it here. See for instance (Li, Chen, and Li 1999). However, the Voronoi diagram only generates an *undirected* graph of corridors. There is still the need to generate a directed graph. This is done by the following algorithm:

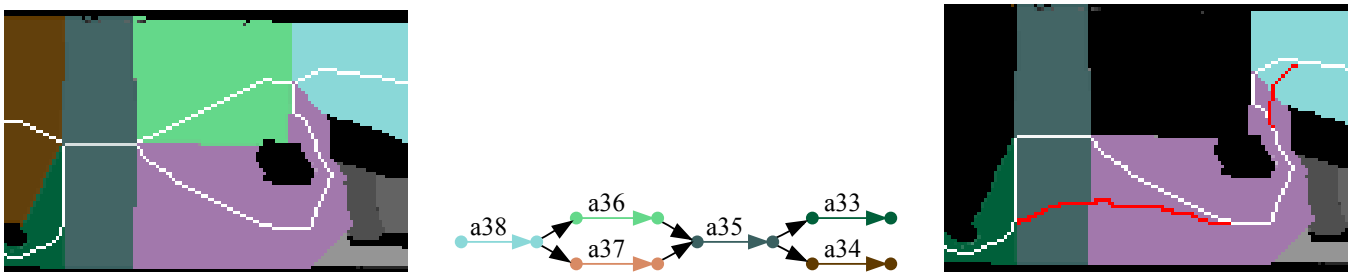

Figure 1: Detail of a corridor plan which gets around an obstacle. The geometrical representation is on the left, with the obstacle in black and the median lines in white. The topological representation is on the middle: the plan dictates that the robot should come from a38 (blue) to a35 through either a36 or a37. On the right, a specific corridor sequence has been chosen for execution by the motion planner and the median line is updated accordingly (red parts).

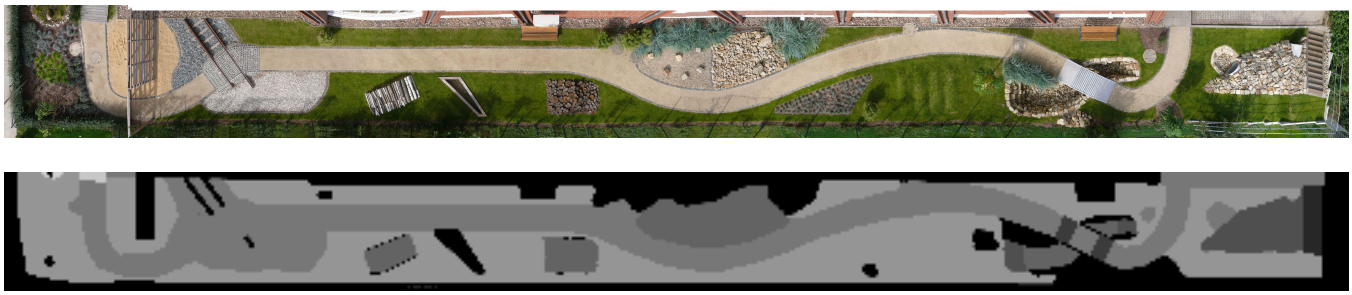

Figure 2: Up: photo of our outdoor test track at DFKI. Down: the hand-made classification per terrain type. Terrain types are internally associated with average speed values. The cost is time to travel.

- the cost along the corridor median lines is made monotonic . Corridors for which it is not the case are split into two new corridors.
- the corridors are oriented w.r.t. cost to target: the entry point whose cost is higher is the entry point of the corridor and the other one the exit point (remember that navigation is following the down-gradient of the navigation function).
- then, a depth-first search starting at the robot's start point allows to find all the paths that reach the goal in a near-optimal way. The cost of traversing a given corridor being the absolute value of the difference between the cost of the entry and exit regions: we make the approximation that the cost of crossing the corridor is equal to the cost of travelling along the median line (per the monotonic transformation in (1)).

The result of this process can be seen on Fig. 3. A detailed explanation, that can help interpreting this plan, is on Fig. 4.

3.3 Choices in the Plan Manager

Our plan manager model is made of two types of objects: *events* and *tasks*. The first represents timepoints and the second represent processes. These objects are structured into graphs (mostly DAGs), to form a plan. These graphs, or *object relations* allow to represent:

- the execution flow by linking events together
- how tasks interact with each other thanks to so-called task relations
- the links between the different levels of abstraction by a mixture of task relations, event relations and an object-oriented model.

One should refer to (Joyeux et al. 2009; Joyeux 2007; 2008) for more details about the model, how the plan manager executes its plans and how it is tailored for multi-robot execution. In this paper, we will only focus on how choices are represented in the plan.

Choices in our plan manager are a special kind of event. That event is present in the execution flow to represent that, once reached, one of the possible plan executions (the event that follow the choice node) has to be chosen (Fig. 5). At all times, if choice nodes are still present in the plans, it means that *at this stage of execution*, there was no compelling reasons to prefer one possible execution path against another.

Finally, the influence of other robot activities on the ability to perform that choice is represented by an *influence* relation borrowed from the GPGP/TAEMS model. This relation marks that choosing one of the outcome tasks is influenced by the task that will provide the necessary information.

4. Discussion

As we mentioned in our introduction, there is no silver bullet in the domain of automated planning, and one has therefore to choose the appropriate planning model and algorithm for

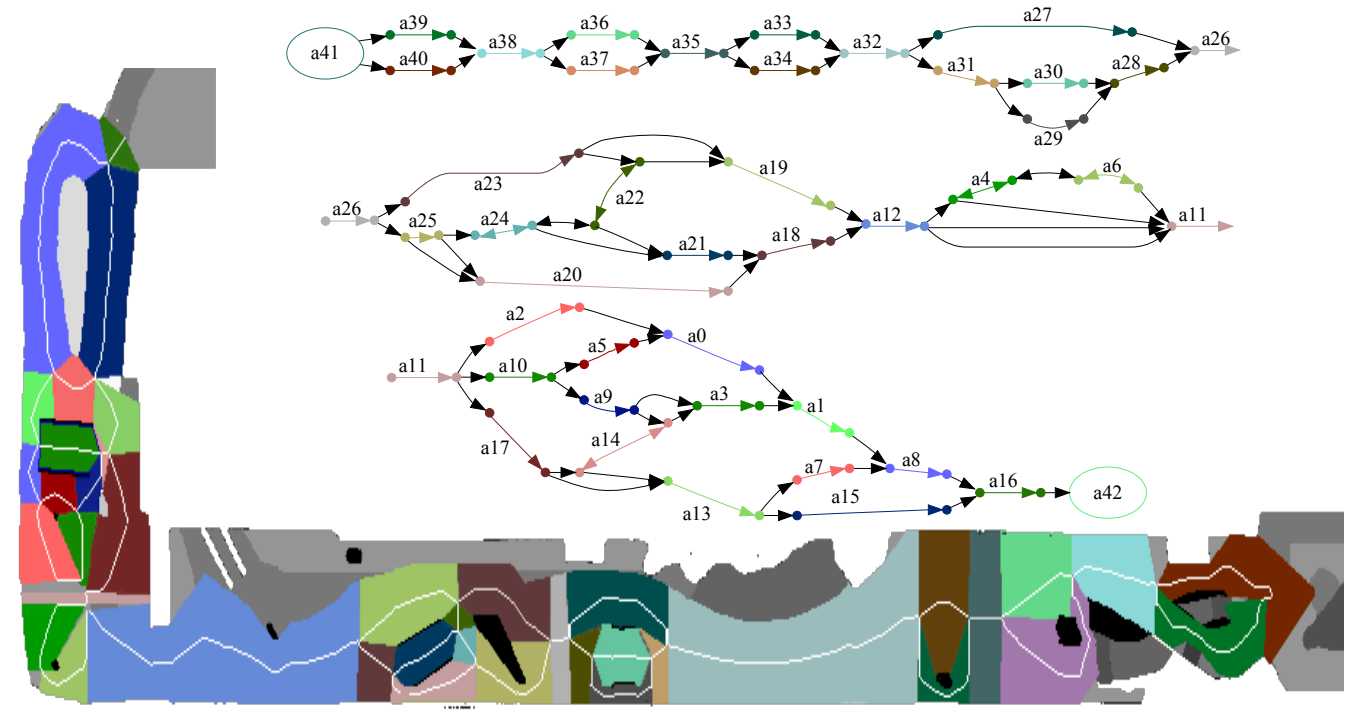

Figure 3: Generated corridor map on the full terrain, for a navigation starting at the bottom far right (on the line between the green and brown regions). The target point is up-left, in the green region. The resulting topological map is presented on the top. Note that each line continues the previous one (for instance a26 is connected to a28 on the right of the first line and to a25 on the left of the second line). The corridor names have no meaning. Detailed explanation of some complex features is presented on Fig. 4.

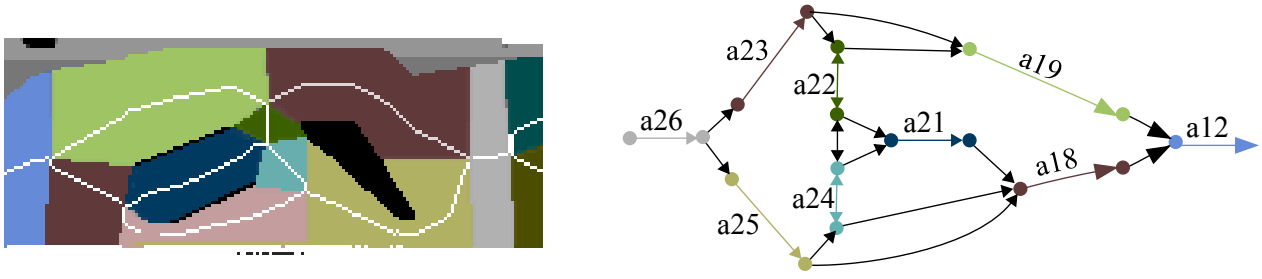

Figure 4: On this detail of the global plan, the concept of bidirectional corridors can be explained. A robot that comes from a23 has first two choices: either go straight to a19 or to go through a22. If it crosses a22, then it has only two choices: either go through a24 and a18 or a21 and a18. It cannot go back a22 because that plan would be ill-formed (see section 3.1). Symmetrically, a robot that comes from a25 can cross both a24 and a22 to go up and reach either a21 or a19. a24 and a22 are therefore bidirectional.

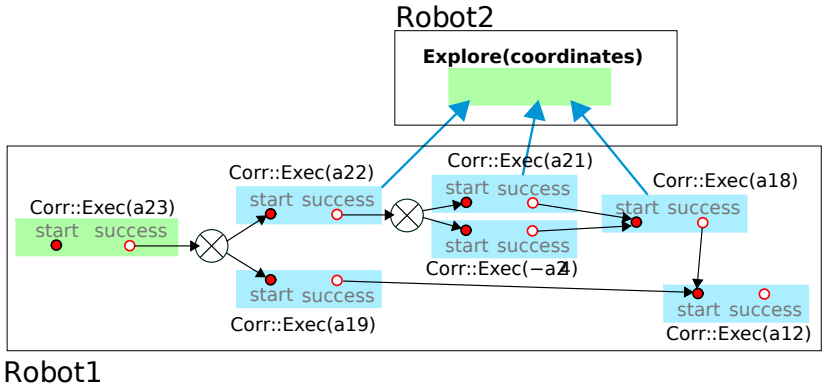

Figure 5: Example plan with choice nodes and influence relation. Parts of the corridor plan of Fig. 4 has been translated into our plan manager's model, the circles being the events, the crossed circles the choice nodes and the event-to-event arrows precedence relations. Then, a hypothetical robot-robot interaction is set up where the second robot influences the first robot's choice by performing some perception task. This relation is represented by the three arrows pointing at Robot2's Explore task (these arrows should be read "parent is influenced by child"). In our manager's scheduler, the choices are removed when done, allowing to simplify the plan as the choices. This is of course a trade-off in itself, as new gathered information cannot modify these choices anymore.

the robot's mission. Obviously, neither is there a silver bullet for the integration of the various planners that a given system needs. The remaining of the paper will discuss the pros and cons of our proposed scheme, and finally conclude on its general usefulness.

4.1 Applicability on the integration of multiple task planners

The choice of task/path planning integration to support our integration scheme is actually biased. Indeed, the path planning domain has some characteristics that make it ideal for our integration method:

- one can find the sensible set of solution efficiently (in our case, using the D* algorithm)
- one can represent this set of solutions in a way that makes it usable for a task planner of plan manager. In this paper, this is through the representation of a single corridor as an atomic task.

We will now discuss the applicability of this scheme in the more general realm of task planning.

First of all, our scheme is related to the domain of least-commitment planning (Weld 1994):

- partial order planners avoid over-constraining the plan at the scheduling level. A single set of tasks is selected (i.e. the planner selects one way of achieving the system's mission(s)), but these tasks are only scheduled as it is strictly required by the plan model. The singular feature of least-commitment planners is to represent disjunctions in the task order, i.e. represent that two tasks A and B that cannot be executed in parallel could be executed either as *AthenB* or as *BthenA*.
- in (Friedman and Weld 1996), action selection is delayed until it is made completely necessary, which reduces the search space. One could also go further and, when there is no compelling reason to choose at the planning level, leave the action selection to the execution component.

Generally speaking, both are a subset of what we propose. Part of the alternative plans would obviously only differ from a scheduling point of view. What could still be added is the ability to find different sets of tasks that achieve the overall robot goal. There is therefore one central question: is it common to have, in a given situation, various ways to achieve a given goal ? Is not partial-order planning the only meaningful part of the "filtering" strategy we propose ?

In our everyday life, most activities would indeed require a fixed set of tasks to be performed. The choices lie only in the scheduling of these activities. It therefore seems that, for most activities, partial order planning would be enough.

However, that changes if we become uncertain about our ability to achieve a task. In automated planning, this is handled in different ways by conditional planning, contingency planning and probabilistic planning. However, in all of these approaches, failure is handled *a posteriori*: the planner takes into account that a task could fail and tries to find a plan, starting from the hypothesized failure. Another strategy would be to postulate that the task *will* fail and ask the planner to find an alternative plan in which the task is not present, or is present in another context. In effect, that would force the planning engine to search for an alternative plan, potentially of higher cost, in which this possible failure is removed. Doing this repeatedly, as contingency planners already do, would lead to the kind of open-choices plans we propose.

4.2 Exploiting Decision Information

The most straightforward use of the decision information is actually static: it can be integrated in a standard planner to take into account concerns of safety (based on how wide and/or well-known a corridor is) and redundancy (how many

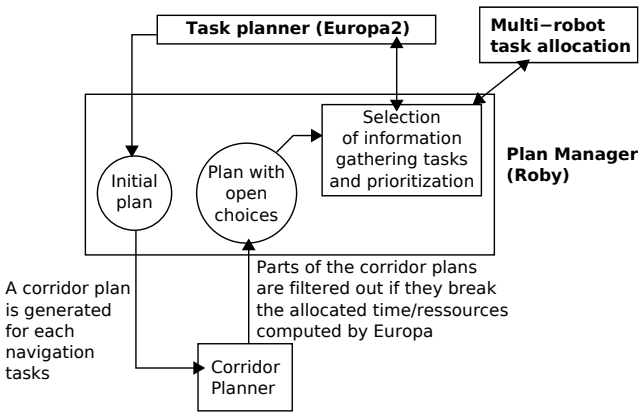

Figure 6: Integration scheme of our path planner with an actual task planner through our plan manager.

alternate paths exist in a subgraph of the corridor plan, to commit the robot as little as possible). This last case would be comparable to the planning process of probabilistic planners, in which the overall expected utility is considered at each level.

Another interesting use of this information is the addition of information gathering tasks, where these tasks provide information that could influence the robot's decision. We plan to develop such a scheme (Fig. 6), basing ourselves on Europa2 as a task planner, the algorithm presented in this paper for path planning and our Roby plan manager. How we would model the influence of one task on the choice node is still an open question for us.

The problem with this scheme is that Europa2 itself will not be able to take into account the corridor plans. The issue is that Europa2's plan database, to our knowledge, does not handle conditional plans. It will therefore not be possible to use the availability of choices in Europa2. We will instead handle it in the plan manager itself. Nonetheless, we assume it will allow us to handle interesting cases, especially in multi-robot contexts.

Finally, based on these corridor plans, we could extend the approach to making contingency plans for navigation. Assuming that it is possible to derive a likeliness that traversing a given corridor fails, it would be possible to build a relationship between both the *nominal* plan and the plan where traversing fails, thus allowing decisions to be taken based on an online estimate of the cost of failure. We already developed the corridor update operator that is needed to implement this approach, but its description is out of this paper's scope.

4.3 Representation of open choices in a planning engine

In our example, the open choices are not represented at the planning engine level, only at the plan management level (i.e. executive), simply because the planning engine is not able to handle this information.

The main reason for that is *efficiency*: handling open possibilities is not possible in the planning stage, because it would make the search cost explode. Nonetheless, we think that the experience in least-commitment planners and contingency planners indicate that our scheme could be integrated in a planning system. Where that would most probably improve the system efficiency is in multi-robot context.

Indeed, multi-robot architectures centered on plan management approaches already indicated (Lesser et al. 2004; Gancet and Lacroix 2005; Joyeux et al. 2009) that it is possible to make single-robot planner or scheduler interact with a component that is multi-robot aware, to lead to an efficient multi-robot system. Our scheme, if integrated in these approaches, would allow the subject of negotiation to not only be task allocation, but a robot's choice of plan. This would be done while keeping the cost functions for single-robot tasks *centered on single-robot concerns*.

5. Conclusion

In this paper, we argued that a richer approach for planner/planner integration is desirable. We used there the path planning as the primary example, because the handling of path plans stemmed our reflexion on the subject and because it is an ideal example for the scheme we propose. We presented how we achieved to build a prototype for a path planner that supports that scheme, and how we plan to integrate it further in a complete robotic system.

Our scheme is based on having planners act a filter instead of decision-makers. In our point of view, one planner should concern itself only with the parameters that are relevant to its level of detail and domain of application, and produce not *the* optimal plan (given its cost function) but the set of plans that are qualitatively equivalent to the optimal one, thus leaving the actual choice to the upper layers (if any).

We also argue that extending it to the integration between task planners is limited, but possible. Limited due to the complexity impact it would have on the planning process, but possible because related approaches already exist. We finally argue that it would significantly improve a robot's ability to cooperate in the context of multi-robot teams.

References

Alami, R., and Botelho, S. 2001. Plan-based multi-robot cooperation. In *Advances in Plan-Based Control of Robotic Agents*, Lecture Notes in Computer Science. Springer.

Alami, R.; Chatila, R.; Fleury, S.; Ghallab, M.; and Ingrand, F. 1998. An architecture for autonomy. *International Journal of Robotics Research* 17(4):315–337.

Beetz, M. 2000. *Concurrent Reactive Plans*. Springer-Verlag.

Finzi, A.; Ingrand, F.; and Muscettola, N. 2004. Robot action planning and execution control. In *Proceedings of IWPSS*.

Friedman, M., and Weld, D. 1996. Least-commitment action selection. In *Proc. 3rd Intl. Conf. on AI Planning Systems*. AAAI Press.

Gancet, J., and Lacroix, S. 2005. Embedding heterogeneous levels of decisional autonomy in multi-robot systems.

Joyeux, S.; Alami, R.; Lacroix, S.; and Philippsen, R. 2009. A plan manager for multi-robot systems. *The International Journal of Robotics Research* 28(2):220.

Joyeux, S.; Lacroix, S.; and Alami, R. 2007. A plan manager for multi-robot systems. In *Proceedings of FSR*.

Joyeux, S. 2007. *A Software Framework for Plan Management and Execution in Robotics: Application to Multi-Robot Systems*. Ph.D. Dissertation, ISAE. `http://tel.archives-ouvertes.fr/tel-00283086/fr/`.

Joyeux, S. 2008. The roby plan manager. `doudou.github.com/roby`.

Lesser, V.; Decker, K.; Wagner, T.; Carver, N.; Garvey, A.; Horling, B.; Neiman, D.; Podorozhny, R.; NagendraPrasad, M.; Raja, A.; Vincent, R.; Xuan, P.; and Zhang, X. 2004. Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. *Autonomous Agents and Multi-Agent Systems* 9(1):87–143.

Li, C.; Chen, J.; and Li, Z. 1999. A raster-based method for computing Voronoi diagrams of spatial objects using dynamic distance transformation. *International Journal of Geographical Information Science* 13.

Muscettola, N.; Dorals, G. A.; Fry, C.; Levinson, R.; and Plaunt, C. 2002. IDEA: Planning at the core of autonomous reactive agents. In *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*.

Nesnas, I.; Simmons, R.; Gaines, D.; Kunz, C.; Diaz-Calderon, A.; Estlin, T.; Madison, R.; Guineau, J.; McHenry, M.; Shu, I.-H.; and Apfelbaum, D. 2005. Claraty: Challenged and steps toward reusable robotic software. *International Journal of Robotic Research* 3(1):23–30.

Thrun, S. 1998. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence* 99(1):21–71.

Volpe, R.; Nesnas, I.; Estlin, T.; Mutz, D.; Petras, R.; and Das, H. 2000. Claraty: Coupled layer architecture for robotic autonomy. Technical report, NASA Jet Propulsion Laboratory.

Volpe, R.; Nesnas, I.; Estlin, T.; Mutz, D.; Petras, R.; and Das, H. 2001. The claraty architecture for robotic autonomy. In *Aerospace Conference*, 121–132.

Weld, D. 1994. An introduction to least commitment planning. *AI magazine* 15(4):27–61.