From: AAAI Technical Report SS-03-04. Compilation copyright © 2003, AAAI (www.aaai.org). All rights reserved.

A Probability–Aware Planning Architecture to Support Variable Autonomy

J. P. Gunderson[†] and **W. N. Martin**[‡]

[†]Gunderson and Gunderson, Inc., Denver, CO
[‡]University of Virginia, Charlottesville, VA
gunders@acm.org , martin@virginia.edu

Abstract

This paper proposes an architecture for autonomous systems that must be deployed in complex domains. The architecture is based on a probability–aware planning and execution system, which allows the system to reason about the uncertainties in the domain, re-evaluate plans during execution and re–plan as needed, and facilitates a non-hierarchical form of variable autonomy which allows functions from high–level, abstract planning, to low–level action selection to be passed to external controllers during plan execution.

1 Autonomous systems don't deal well with confusion

While significant strides have been made over the last 30 years in using robotics for factory and industrial automation, the use of these systems in more complex environments has lagged behind. One characteristic of factory environments is that extreme steps are taken to remove any uncertainty or variability in the workspace. This simplifies the software and reduces the need for error or exception handling. When autonomous systems are deployed into environments that are only partially engineered, current systems must rely on guidance and assistance from external sources. The ability to vary the level of autonomy (also called traded control, and adjustable autonomy) with either external control systems with more resources, or with humans seems to be critical.

The use of autonomous systems in complex environments requires complex, cooperative interaction between the autonomous system and the humans in the environment. This interaction can be problematic, and can lead to dangerous situations. The development of autonomous air vehicles was intended as a force multiplier, as well as a mechanism to reduce the risks to the modern warfighter. One deployed model originally required 8 full time support personel and over the intervening years this number has been reduced to only five. This still requires detailed, exact communication between the UAV and the controllers, and transitions between different levels of autonomous behavior on the part of the vehicles.

The autonomous system must be capable of accepting guidance (goals) from the human partner, quickly produce appropriate courses of action to assist in acheiving those goals, adapt to changes in the levels of autonomy available, and reformulate those courses of action in response to changes in the environment and in the goals themselves.

This latter aspect, the ability to adjust to a dynamic environment and to a changing mix of goals is addressed by a probability-aware planning and execution architecture, presented in this paper.

2 Definitions

So what are we talking about here? What is a complex environment? What does robust intelligence mean? And what actually varies under variable autonomy? John Von Neumann once said "There's no sense being exact about something if you don't even know what you're talking about," so first we should specify what we're talking about.

Much of the research into autonomous systems has focused on environments which are inherently simple. The early planning research was based on three assumptions:

1. **Omniscience:** the autonomous system knew everything about the environment at all times;
2. **Determinism:** whatever actions the system applied were guaranteed to succeed; and
3. **Stability:** the state of the world was constant, unless the autonomous system altered it.

Unfortunately, none of these hold in a complex environment, in fact the inverse of these assumptions almost define the characteristics of a complex environment.

This means that if an autonomous system is going to succeed in these environments, it must be designed to deal

with incomplete or inaccurate information about the world, it must be able to reason about the possible unintended consequences of failed attempts to make things better; and it must be able to deal with exogenous events - changes to the state of the world, changes to goals, and changes to the mechanisms available to achieve those goals.

Robust intelligence is fairly straight forward: we define it as the ability to achieve goals in a dynamic and uncertain domain[13]. Since the domains are dynamic and uncertain, achieving a goal once says nothing about the autonomous systems ability to achieve goals in general; the metric used is the probability of goal satisfaction. In general, for a given target specification:

$$T \equiv (F, A, i, g)$$

where:

F is a set of Features of the domain,
A is a set of Actions
$i \in F$ is an intial state; and
$g \in F$ is the goal state.

and the result of solving the problem is a sequence of Actions applied to the world to transform i into g.

A key point of the *Robust* aspect is that it is not sufficient to produce a single sequence of actions to achieve g, As a sequence is applied to the world, the outcomes of the actions are not certain, and the world is not stable, so the probability that the initial sequence will achieve g is small. When the current actions fail, the autonomous system must be able to develop a new sequence that will transform the new world state i' into the goal state g.

Variable autonomy has been discussed in numerous papers[12, 15, 2, 20], and as the topic of several workshops and symposia. In essence, variable autonomy is the ability of an autonomous system to surrender some of its decision making or action application ability to an external controller. Thus, if an autonomous system is capable of effectively executing a sequence of actions, but it cannot produce the correct sequence with its internal resources, it could accept an externally genetated sequence and then execute it. Alternatively, the autonomous system might be perfectly capable of deciding what actions need to be done to achieve a goal, but lack the descrimination in sensors or effectors to reliably execute the actions. In this case the variable autonomy would be to surrender the low-level execution decisions to an external controller.

In both of these cases, the autonomous system must manage a smooth transition from one arrangement of control flow between modules and another arrangement in which some of those modules are external to the autonomous system. In turn, the system must be capable of negotiation a transition back to an arrangement in which all decisions are local.

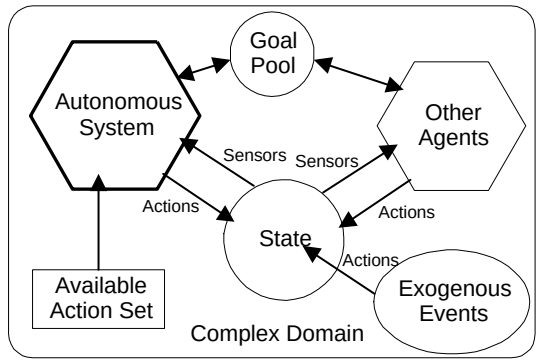

Figure 1: Complex Domain - The autonomous system must deal with a dynamic domain, with uncertainty in its sensors, other agents making inexplicable changes, and probabilistic action that may not achieve their intended effects.

3 Requirements for Success

If an autonomous system is deployed into a complex environment it must be able to respond to changes in the environment caused by numerous sources. Some of the sources shown in Figure 1 include the stochastic nature of the domain itself (exogenous events), other agents in the domain (some of which may be human), the changing nature of the goals (goal pool), and the probabilistic effects associated with the actions.

This leads to a set of requirements that an autonomous system must meet if it is to be effective:

- It must be able to accept a range of goals and generate behaviors to achieve them;
- It must be able to reason about the probabilistic nature of the behaviors;
- It must be able to monitor the execution of the actions, and detect their failure;
- It must be able to detect externally caused changes to the environment, and reason about the impact of those changes on its plans; and
- It must be able to transfer responsibility for aspects of the goal satisfaction to external controllers.

4 Architecture

The architecture is the Probability–Aware Planning and Execution System. This architecture has been designed to meet the five requirements above, and has been shown to be a robust autonomous system. The general control flow is shown

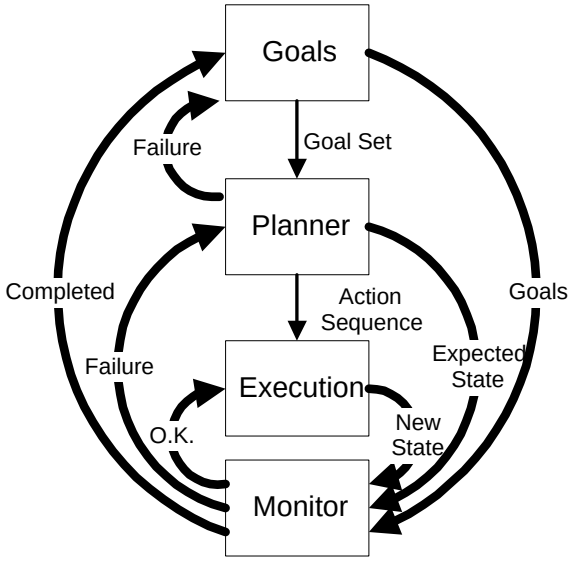

Figure 2: System Architecture - The architecture is an interleaved planning and execution system, which continoually monitors changes in the goals pool, selects goals, implements plans to achieve those goals, and then executes the actions. The execution is also monitored for either action failures or detected changes to the environment that might disrupt future actions.

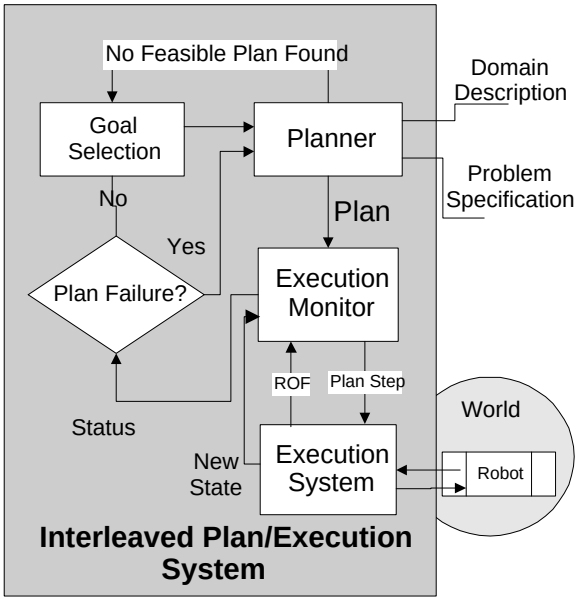

Figure 3: System Design - the architecture is broken down into four primary modules: the goal selector, the planner, the execution monitor and the execution subsystem. The last of these communicates with either the robot in the complex domain, or with an external simulator using the same interface. The connection uses Java's RMI mechanism.

in Figure 2. The outer loop of the system is monitoring the set of goals in the goal pool, and compares these with the perceived state of the world. If they are consistent no further action needs to be taken. However, if either a new goal is inserted into the goal pool, or the world state changes to one that is inconsistent with the goals, a set of goals (possibly incomplete) are selected for achievement, and the system enters its planning mode.

In the planning mode, a sequence of actions is selected that, if applied successfully, will achieve the goals. This planning system can be a single level end-to-end planner, or can be a multi-resolution planning system[1], however it must be able to reason about the dynamism and the uncertainty associated with the domain. During this mode, the system may determine that it does not have the resources to achieve the current goals. This could be due to either not having required actions in its action set, or may be due to excess computational requirements. In either case, it can either accept partial goal satisfaction or choose to return control to the goal selection mode.

Once the action sequence is developed, it and the expected world states during execution can be passed to the Execution system. The expected world states are developed during the planning process, and are used to monitor the execution of the plan. After each action is executed, and at any point during execution, the systems sensors can be queried to see if the perceived world is consistent with the expected world. If all is as expected, execution continues by confirming that the world is consistent with the preconditions of the next action, and that the goals have not been met, and then emitting the next action.

In the event that the moniotor detects an execution failure, control can be progressively returned to Execution mode for retry on failure, the planning mode for the development of a new action sequence, or to the goal selection mode for regoaling.

5 Implementation

The planning and execution system is implemented in Java, and is designed for either single processor or distributed processing. The interleaved planning and execution system is depicted in Figure 3, which shows the interconnections and control flow. In the general case any planner can be used, if it supports a common interface, and can accept Java Remote Method Invocation (RMI) requests. This can be accomplished by placing a non-compliant planner in an appropriate wrapper.

In the existing implementation the planner is a probability-aware planner, which reasons about the likelihood of goal satisfaction during the planning process. It does not require

a complete model of all possible outcomes of actions, and it does not attempt to completely model the interactions of alternate outcomes. Hence, it is not a probabilistic planner[4, 7, 16], it is only probability-aware.

5.1 Probability–Aware Planner

A core characteristic of complex domains is that they are dynamic. A factory floor with industrial automation is engineered to remove any and all unplanned events, and in these domains autonomous systems thrive. However in domains with human actions, non-coordinated actions by multiple agents, and stochastic events; different techniques are required to achieve goals. Numerous planning systems have been developed to model uncertainty in actions. These include Decision Theoretic planners, modeling uncertainty in the world using partially observable Markov Decision Planners (POMDPs), and systems have been developed to respond to the effects of exogenous events[5]. However, the complexity issues have been daunting.

The approach used by the Probability–aware planner is derived from research in cognitive science, and accepts the fact that computational resources are limited, and time is often short[10, 18, 17]. Rather than attempting to produce a single plan which is optimal in some sense, the approach is to come up with a plan that is good enough, and when it fails, replan. Previous research has indicated that in uncertain and dynamic domains there is an effective planning horizon, beyond which improving the plan can decrease the effectiveness[1] of the system[6, 14].

The input to the planner is the feature set, the action set, and the initial and goal states. The action set is more complex than many planning systems since it must encode the probabilistic nature of the actions. Each action has a guard clause (pre-conditions which must be true for the action to be valid) and one or more outcomes. Each outcome specifies the changes to the world that it causes, and the probability that this outcome will occur. In general, there is an 'intended' outcome, which is the most probable; however this is not mandatory. In addition, there can be several alternative outcomes, each with its own probability of occurence.

The probability–aware planner produces a set of candidate plans by performing a forward chaining search of the plan space. During this initial planning phase, only the most probable outcome is evaluated (optimistic evaluation), and the planner calculates the cumulative probability of success of each plan that is expanded. This builds a planning graph by expanding nodes in a highest *a priori* probability order (See Figure 4). Those plans which achieve the goals are added

[1] The focus is not on producing the 'best' single plan, but rather on maximizing goal satisfaction. Resources applied to optimizing a plan in in World i, might preclude the solution of a problem when the world changes to state i', and reduce effectiveness

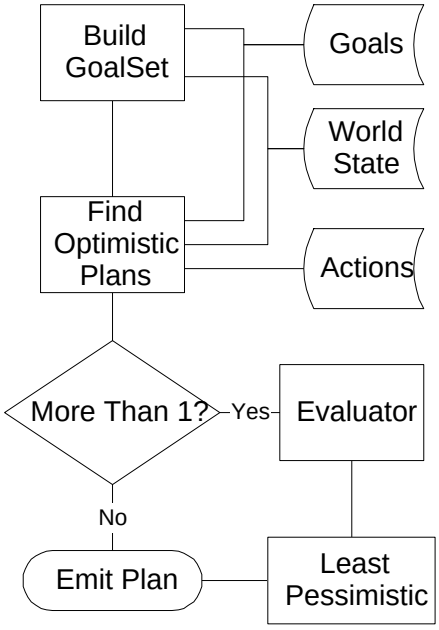

Figure 4: Internal Planner Architecture - The basic model is: build the set of goals to be met; expand a plan graph using best-first on the *a priori* probability
; select the top N plans that achieve the goals, based on optimistic execution; re-order the candidate plans using Monte Carlo simulation of possible failures; select the best plan.

to the set of candidate plans. After either a user defined number of candidates have been generated, or the planner exceeds a user defined search limit, the candidate plans are simulated using a Monte Carlo simulation to capture plan step interactions[11]. During the simulation phase, the alternate outcomes are evaluated according to their probability distributions, and overall plan success rates are determined. The candidate plan with the highest simulated success rate is selected for execution.

Unfortunately, while forward chaining planners have been shown to function well in probabilistic domains, they often are computationally intensive. To address this some form of hierarchical planning is frequently applied.

5.2 Multi-Resolution Planning

In the Probability-aware planner, a planning problem is specified by a set of world features and allowed ranges, a set of action outcomes with associated probabilities, an initial world state, and a goal world state. Semantically the actions are terminal events that cause a transition from one world state to another. If these actions actually reflect the ground actions in the domain, then plans can be extremely long, and the planning process can be very computationally intensive. In a complex domain this is not good.

One approach to reducing the computational burden is to break the complete planning problem down into a set of smaller problems, either by finding midpoints in the plan, or by using aggregated operators, planning in abstraction spaces, or using hierarchical task networks, a related approach by Meystel and Albus is Multi-Resolution Planning.

A traditional planning system has a set of actions available to the planner. The planner uses these to construct a sequence (either totally ordered or partially ordered) of transitions, which, when applied to the intial state, cause changes to the world such that the world is in the goal state. These actions are grounded, in that they can be directly executed in the world. Abstract planning does essentially the same thing; however, the actions are not grounded, rather they represent combinations of less abstract actions. Eventually, all abstract action sequences can be reduced to a sequence of grounded actions.

In many abstract planning systems, the decomposition of an abstract action is done by following a specified pattern such as macro substitution (MACROPS in the abstract STRIPS approach[9], or task networks[19] in NOAH and subsequent HTN planners[8]). The approach we are developing instead relies on the ability of the Planning and Execution system to invoke different planners at different times (See Figure 5). In this model, grounded actions are designated as directly executable, while abstract actions specify a planner and a domain. When the system develops an abstract plan at the highest level, the plan can consist of a mix of grounded and abstract actions.

Each abstract action in the current plan is presented as a new planning problem (with its own action set, inital state and goal state) to its designated planner, and a sub-sequence of actions that satisfy the new planning problem is returned and joined into the overall plan. This of course might lead to sub-goal interactions which could result in either sub-optimal or infeasible plans[3]. Those plans which are infeasible are easily removed from the set of candidate plans during the Monte Carlo phase of the planning process, and sub-goal interactions that affect the probability of success of the plan will alter the ordering of the candidate plan set, such that the plan with the highest simulated success rate is selected for execution.

Since planning is occuring at multiple levels from highly abstract 'big–picture' plans, to low-level 'how to best get down the hallway' plans, the architecture provides a non-hierarchical variable autonomy capability.

5.3 Variable Autonomy

Effectively, every one of the planners is an external application which has a specific interface. It is given a world description to start from, and a goal, and has a set of operators that it can apply to achieve that goal. It returns a sequence

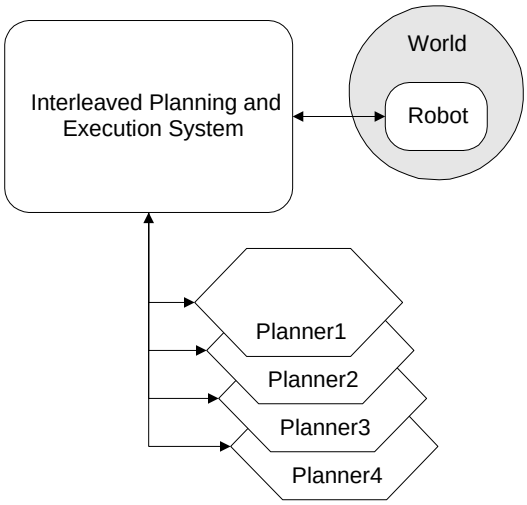

Figure 5: Multi-Resolution Planning -

of actions (either abstract or grounded or a mixture of the two) which have a probability of achieving the goal. Clearly, any applictaion which can accept that input and return a valid output can be invoked. Including a person on the other end of an interface.

The only necessary component is the ability of the planning and execution system to request support, and to be able to respond to a request to surrender control over part of the task. Given that mechanism one can easily envision a cooperative session between autonomous robots and humans in which, if a robot cannot find a high–level solution to a problem, a human operator can provide a high level plan which the robot expands and executes. Alternatively, if the robot has the high-level plan, but cannot find a sufficiently reliable way to execute it, the plan could be passed to the human, and the robot could be 'walked through' the plan, using the humans senses and dexterity to supplement its own.

Under development is an extension to the probability–aware planning and executions system which will support the trade-off of control at any planning resolution from abstract to individual action emmission. The ability of the Planning system to reflexively monitor its progress towards the elaboration of a sufficiently successful plan at any planning level is coupled with the ability to proactively terminate a planning process and substitute an external control system during the planning process is critical.

However, deploying autonomous systems into complex environments, and into environments where they must hold up their end of the human interaction requires that the five criteria identified in Section 3 be supported by the architecture of the autonomous system. The ability to maintain and act on a range of goals is supported by the high-level goal selection process, and the cyclic process of re-evaluating old goals,

and accepting new goals. The ability to reason about the probabilistic nature of the world and actions in the world is supported by probability–aware planning. The embedding of the planning and execution into an "observe, orient, decide, act" loop, supports the ability to monitor, detect and react to failures. The inclusion of the observation ability within the loop structure provides the ability to detect and respond to exogenous events in the domain, and the architectural integration of a mechanism to allow the invocation of an appropriate control mechanism, during the execution of a plan enables the realtively seemless integration and trade-off of control to external control systems.

References

[1] J. Albus, A. Lacaze, and A. Meystel. "multiresolutional intelligent controller for baby-robot. In *Proc. of the 10th IEEE Int'l Symposium on Intelligent Control*, Monterey, CA, 1995.

[2] K. S. Barber and C. E. Martin. Agent autonomy: Specification, measurement, and dynamic adjustment. In *Workshop on Autonomy Control Software, Agents-99*, 1999.

[3] A. Barrett and D. S. Weld. Characterizing subgoal interactions for planning. In *Proc. of the nth International Joint Conference on Artificial Intelligence*, pages 1388–1393, 1993.

[4] Avrim Blum and John Langford. Probabilistic planning in the graphplan framework. In *ECP*, pages 319–332, 1999.

[5] Jim Blythe. Planning with external events. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–94)*, pages 94–101, San Francisco, CA, 1994. Morgan Kaufmann Publishers.

[6] M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource bounded practical reasoning. *Computational Intelligence*, 4(4):349–355, 1988.

[7] D. Draper, S. Hanks, and D. Weld. Probabilistic planning with information gathering and contingent execution. In *International Conference on Artificial Intelligence Planning and Scheduling*, 1994.

[8] K. Erol, J. Hendler, and D. S. Nau. HTN planning: Complexity and expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, volume 2, pages 1123–1128, Seattle, Washington, USA, 1994. AAAI Press/MIT Press.

[9] R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4):251–288, 1972.

[10] G. Gigerenzer, P. M. Todd, and the ABC Research Group. *Simple Heuristics that Make Us Smart*. Oxford University Press, 1999.

[11] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*, chapter 1. Chapman and Hall/CRC, 1996.

[12] J. P. Gunderson. Effects of uncertainty on variable autonomy in maintenance robots. In *Workshop on Autonomy control software*. Autonomous Agents '99, 1999.

[13] J. P. Gunderson and L. F. Gunderson. Effective planning horizons. In *Proceedings of 3rd Annual Workshop on Performance Metrics for Intelligent systems, PerMIS2002*, pages XXX–XXX. PerMIS, 2002.

[14] J. P. Gunderson and W. N. Martin. Effective planning horizons for robust autonomy. In *American Association for Artificial Intelligence Spring Symposium Series*. American Association for Artificial Intelligence, 2001.

[15] D. Kortenkamp, R. P. Bonasso, D. Ryan, and D. Schreckenghost. Traded control with autonomous robots as mixed initiative interaction. In *AAAI Spring Symposium on Mixed Initiative Interaction*, 1997.

[16] N. Kushmerick, S. Hanks, and D. S. Weld. An algorithm for probabalistic planning. *Artificial Intelligence*, 76(1–2):239–286, Jul 1995.

[17] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

[18] Allen Newell and Herbert A. Simon. *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, USA, 1972.

[19] E. D. Sacerdoti. The non-linear nature of plans. In *IJCAI-75*, 1975.

[20] G. S. Wasson, J. P. Gunderson, B. S. Graves, and R. Felder. Effective shared control in cooperative mobility aids. In *FLAIRS-2001*, pages 509–513, 2001.