Effective Planning Horizons for Robust Autonomy

J. P. Gunderson and W. N. Martin

University of Virginia, Dept. of Computer Science
Thornton Hall
Charlottesville, Virginia 22903
gunders@acm.org

Abstract

Robust autonomy, in the sense of performing tasks in the face of dynamic changes to the environment, requires that an autonomous system be capable of responding appropriately to such changes. One such response is to effectively adapt the allocation of resources from planning to execution. By adapting the resource allocation between deliberation and execution, an autonomous system can produce shorter plans more frequently in environments with high levels of uncertainty, while producing longer, more complex plans when the environment offers the opportunity to successfully execute complex plans.

In this paper we propose the idea of the "effective planning horizon" which adapts to environmental changes to bound the deliberation in an interleaved planning/execution system. The effective planning horizon is developed from an analysis of the advantages and disadvantages of three classic autonomous system architectures as feedback control systems. This leads to the development of an analytic model which suggests the use of maximizing the expected value of plans by adjusting the planning horizon.

Introduction

If autonomous systems are to be deployed outside of research institutions, and if they are to perform useful functions, they must be able to continue to function in the face of uncertainty and in dynamic environments. There have been numerous approaches to providing robust autonomy, and by examining some of these approaches, we can extract some general features that can be used to categorize autonomous systems.

For the purposes of this analysis we evaluate 'robust autonomy' in the following sense: *the ability of an autonomous system to continue to satisfy goals by adapting its behavior in response to changes in the environment.* Using this definition, robust autonomy would include the ability to cope with failures such as damaged sensors or effectors, the ability to adapt to changes in the rates of exogenous events in the environment, and to modify behaviors in response to changing mission demands on the system.

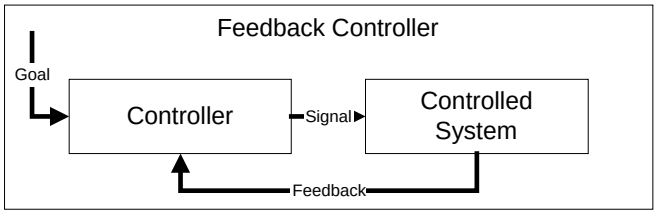

Figure 1 Feedback Control Model

Feedback control models are a mature mechanism for providing robust control in dynamic environments. These models address goal maintenance as a process of sensing the current distance from the desired state, and producing a corrective signal, with the correct magnitude and direction to bring the system to the goal state in optimal time. We use this framework to analyze common architectures of autonomous systems, and propose the effective planning horizon model as a mechanism which can be used to provide robust goal satisfaction for autonomous planning systems

Feedback Control Model

One traditional approach to goal satisfaction under uncertainty is by closed-loop control (See Figure 1). The system consists of two main sub-components: a controller, and a controlled system, connected in a feedback loop. In one phase, the state of the controlled system and the environment is sensed, and if the system is not in its desired goal state, a corrective action is selected. This corrective action is passed as a control signal into the execution phase, where the controlled system applies the action, and affects the environment. The effect of this change is sensed and the results are passed back to the controller, which closes the loop. Since this system is constantly sensing the changes to the environment and comparing them with the desired state, it is very robust with respect to uncertainty in the sensors, actuators, and external events in the environment. One critical feature of these feedback control systems is the gain of the controller. Clearly, if the controller does not react

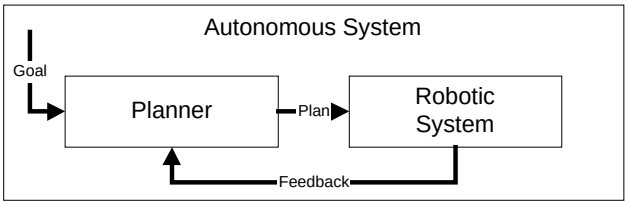

Figure 2 Autonomous system as a feedback controller

sufficiently to changes in the controlled system, the goals will not be met, however if the system is too responsive, it over-reacts to small fluctuations and the goals are also not met.

This type of system is very common in robotics. It is used in servomotors, where the motor tries to minimize the difference between its current orientation and the control signal; video servoing is frequently used in industrial robotics to precisely position effectors with respect to a part; and drive motors frequently use encoder feedback to maintain a constant velocity across varying terrain.

Multi-tier architectures as feedback controllers

Another way to look at the feedback control model is that the controller uses a model of the world state, updated by sensors, and a goal state and produces an output that is intended to change the world from its current state to the desired state. While these controllers are traditionally used to maintain the goal state, during startup they must first achieve the goal.

This corresponds very closely to the traditional model of a planning system, which uses a model of the current world state and the desired world state, and produces a plan (or sequence of changes) to achieve the goal. Thus a planning system can be represented by Figure 2. The typical difference between a planner and a feedback controller is that the planner produces a single plan to achieve a single goal, rather than a sequence of plans to maintain a state. However, in deployed systems, which must be able to handle uncertain and dynamic environments, hybrid architectures have been developed.

Hybrid System Architecture

One of the most popular architectures for autonomous systems is a layered, hybrid system. These systems are popular because they work, that is to say they provide a robust architecture for autonomous systems (Simmons 1994). One characteristic of multi-layered architectures is that they interleave the deliberative process with the execution of actions. In this model a period of deliberation occurs in which the system produces a plan to achieve its goals. This plan may take the form of a partially ordered sequence of actions, a selection and ordering of skills, or a structuring of behaviors, but the system has made choices with the intent to satisfy a goal

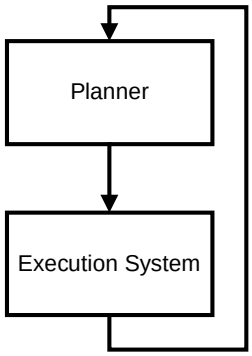

Figure 3 Multi-tier Autonomous system architecture, reduce to two layers, planning and execution

or goals. Once this plan is developed, it is put into effect, and the autonomous system attempts to change the world from its current unsatisfactory state into a world state that is preferred.

However, it has been said that “No plan survives contact with the enemy”, and in a dynamic and uncertain world, no plan is guaranteed to succeed. As a result, steps in the plan may fail, or the skills or behaviors selected may not be sufficient to achieve the goals. In this case, if the system is to be robust, it must achieve goals in spite of these failures, and so, the deliberation is again undertaken, a new plan is formed and the system tries again as in Figure 3.

Interleaved Planning and Execution

So, one might argue that every autonomous system is an interleaved planning/execution system. This is clear for a traditional, multi-tier system. But is less straightforward for both reactive, behavior-based systems, and for classical planners.

Classical Planning

Classical planners, such as the original STRIPS system (Fikes and Nilsson 1971), were given a world model, and operator set, and a goal state, and produced a plan to achieve the goal. This classical planning paradigm has been implemented in numerous other deployed systems.

For a classical, task-based planner, the sequence is:

1. Input the goal state and the current world state
2. Produce a plan which achieves the goal
3. Output the plan

This sequence is iterated once, and it is assumed that the world model is accurate, the execution is flawless, and no other events occur during planning and execution. After this, the planner is shut down, and the next planning

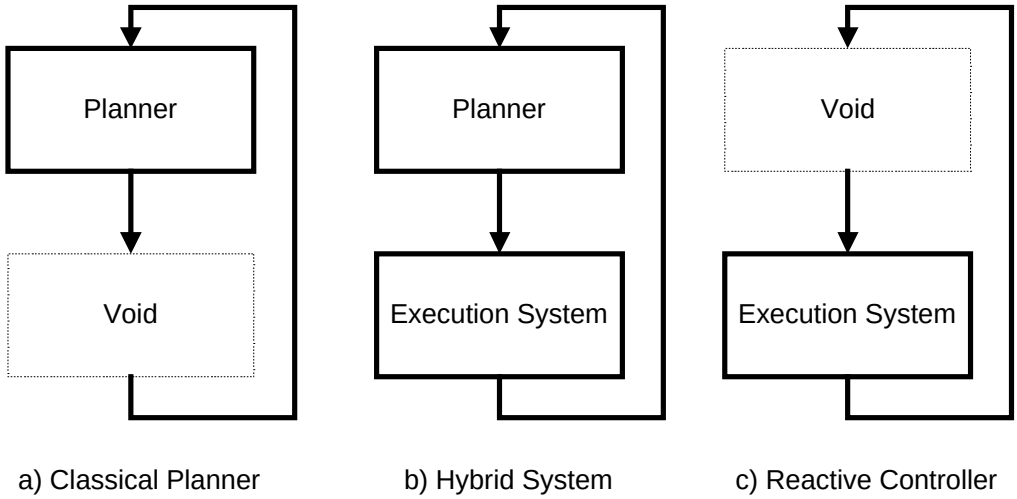

Figure 4 Comparative models for three common planning systems

episode occurs with no reference to the previous sequence.

After the goal is achieved, a new goal is proposed, and the planning stage is entered again, to begin the next iteration. However, the planner is not scrapped after a single planning episode, it is used again and again, sometimes on very different domains, and sometimes on very similar domains. Thus we can represent a traditional planner as an interleaved planning/execution system with an empty or void execution phase as in Figure 4a. It is repeatedly invoked to produce plans, and simply assumes that the execution will take place flawlessly.

Reactive, Behavior Based System

In the case of a 'hard-coded' reactive system, the controller senses the environment, chooses an action, and executes it. The resulting world is sensed, and the process repeats, with the controller selecting the best action for each sensed world state. These actions are often modeled as competing behaviors in a subsumption architecture (Brooks 1986). This system most closely resembles a pure feedback controller, with one important difference: in a typical feedback controller the complex relationship between conditions and actions are modeled as continuous functions (differential equations, usually) and the effects of each action are precisely captured. This allows the controller to select precisely the correct type and amount of change to apply to the world to return to the goal state. Unfortunately, we have no such differential equations to capture the effects of behaviors in the general planning problem. Thus the developer frequently uses *ad hoc* heuristics to establish the precedence of the competing behaviors and adjusts these to achieve each particular goal.

With reactive systems, one question is: who does the deliberation? Is the reactive system deliberating when it selects an action based on its percieved world state, or was the deliberation done when the behaviors and skills were assembled and prioritized. Under the latter view, the planning phase is void (or, perhaps it occurs once when the behaviors are hand crafted for the current task) and the execution phase is repeatedly invoked (Figure 4c).

It might be argued that the production of an optimal behavior structure will do the best possible in the domain, and so no 're-planning' is required. However, one can argue that an empty or void planning stage occurs, in which the same mix of skills or behaviors is produced, and the plan is re-executed.

Common Framework

What do we gain by adding these void stages? One gain is that we can place these models into a space in which they can be compared. We can also examine why these characteristics are present in these systems, and suggest additional ways to benefit from them.

Why does the architecture of a traditional planner assume that the plan will execute flawlessly? In part this is due the historical development of deliberative planners from theorem provers, and in part it is due to the partitioning of the goal satisfaction problem into two disjoint phases: first figure out what to do, then do that thing. Classical planning made the assumption that if we only knew what to do, the problem would be solved: planning was the hard part.

Unfortunately, it turned out that the 'doing' was equally difficult. Significantly before the earliest work on General Problem Solvers (Newell and Simon 1963) was going on, purely reactive systems were being developed by W. Grey Walter (Walter 1950). The turtles developed in the 1950s were hard wired, vacuum tube based, mobile robots, which wandered down hallways, avoided light or dark, and in some cases sought out energy sources to recharge themselves. These systems had no deliberative layer,

other than the design of their hardwired circuits, and simply reacted to their environments by enabling and inhibiting behaviors, an early foreshadowing of Brooks subsumption architecture.

Modern hybrid systems have evolved out of the failings of both of these earlier architectures. An autonomous system cannot assume the flawless execution of the plan produced by its deliberation, nor can a purely reactive system easily achieve complex goals by a single set of interacting behaviors. Hybrid systems attempt to solve the problem by joining the two approaches together. The result is a loose coupling of deliberation and reaction, with very different approaches by each layer.

Robust Autonomy

It has been long accepted that as uncertainty and dynamism increase in an environment, the greater the need for more reactive systems. In recent work (Gunderson and Martin 2000) quantitative relationships have been suggested for the impact of three types of domain uncertainty on the ability of an autonomous system to achieve goals.

This work classifies domain uncertainty into three categories:

1. Sensor uncertainty,
2. Effector uncertainty, and
3. Exogenous events.

These are used in a simulation system that measures the goal satisfaction of a simulated maintenance robot under widely varying levels of each of these types of uncertainty. The results suggest that autonomous systems are very sensitive to even low levels of uncertainty in the environment, with overall goal satisfaction dropping to 75% with the introduction of 10% sensor errors. In addition, this research reported that the ability to retry on failure was very effective at maintaining goal satisfaction in the face of all types of uncertainty. The simulated robot was more robust (i.e., able to satisfy goals as the environment changed) when it used different deliberation and execution mixes in response to different levels of uncertainty in the environment.

Adapting to failure

Our recent work has focused on the use of probability-aware planning systems, which actively update the probability of success of possible operators to produce the "Plan most likely to succeed," rather than the shortest feasible plan. These systems use lightweight uncertainty models to provide key insights into selecting between alternative feasible plans based on predicted plan success. In addition, these systems function as integrated planning/execution systems, with low-level support for operator probability values, and adaptive goal prioritization (Gunderson 2000). This gives the planning system the ability to adapt to failures in its environment, including adapting to mechanical breakdown of sensors and effectors.

However, being able to function effectively in the designed environment may not be sufficient for a deployed autonomous system. There will be errors in any characterization of an environment, and environments change. If an autonomous system is to be robust in the face of changes, it must be able to adapt. The system must have a 'knob to twist' to respond to change.

The three architectures depicted in Figure 4 suggest that such a knob exists: turned to one extreme, it provides a classical planner, turned to the other extreme it produces a purely reactive system, and in between it provides a hybrid system with greater or lesser proportions of planning and reacting. But what is the scale associated with the knob?

Planning Horizon

One way to establish the scale is to look at the characteristics of the three systems. Table 1 compares two salient features of these three: plan length and number of executions. In the table, N is used to indicate that the feature can be as large as needed, while K indicates a fixed number.

A classical planner, produces a plan as long as needed to satisfy the goal (size N), and provides a single execution. A reactive system issues single operation plans (or enables a single behavior) but does so as many times as needed to achieve its goal (size N). The hybrid system issues plans of varying length, and executes them until they complete or until a failure occurs, and then repeats the process as needed. The plan length is frequently bounded to control the exponential growth of the planning process (plan length K), and the execution is often bounded as well: "try three times, then fail" is a common technique, hence size K. This suggests that the knob we are looking for is the length of the plan considered.

Table 1 Comparison of Architecture characteristics

	CLASSICAL	HYBRID	REACTIVE
Plan Length	N	K	1
Execution Cycles	1	K	N

It is commonly accepted that as the levels of uncertainty and dynamism in the environment increase, the value of long plans decreases. In effect, the value of each plan step is discounted by the likelihood that the world state will meet its preconditions at the time it is executed (Bratman, Israel and Pollack 1988). So our system can adapt to changes in the environment by adjusting is planning horizon to be effective.

Control Theory Revisited

By reducing the length of the plans that are considered, the planning process speeds up. In addition, shorter plans execute more swiftly, which has the effect of shortening

the planning/execution cycle. One effect of this shorter planning execution cycle is that the autonomous system can apply more corrections (execute more plans) in the same amount of time. If there is no uncertainty, this has no real impact, since presumably the first plan will achieve the goal. However, when the plans are being executed in an uncertain and dynamic domain, the first plan may fail, or only achieve some of the goals needed to achieve the task. In this case, the ability to develop and execute a second plan increases the robustness of the system, since only if all the plans fail will the goal be unsatisfied. This means that as the plan length decreases the system is more reactive to uncertainty in the environment. So why not reduce the plan length to one, and always run in a reactive mode?

Using the feedback control model, we know that there is a penalty associated with a control loop that is under damped (Franklin, Powell, and Emami-Naeini 1994). But does this carry over into the deployed robot domain? In recent work (Kaga *et al* 2000) on fault tolerance in mobile robot based target tracking, the results indicate that with complex tasks reacting too quickly does, in fact, lower the quality of the solution. Clearly, dependent on the uncertainty in the environment, there is an "effective planning horizon": a bounded plan length that produces robust behavior.

Effective Planning Horizon

Our current work focuses on determining empiric relationships between levels of uncertainty in the environment and the effective planning horizon. The current model suggests that the effective planning horizon is the plan length that maximizes the expected value of the plan. To develop this expected value, we use the following assumptions:

1. Preconditions make plan steps conditionally independent.
2. The levels of uncertainty in the environment determine the probability of success of each plan step.
3. Satisfying some of the goals of the system is better than not satisfying any (partial goal satisfaction)
4. Plan steps are 'fail negative', in that they either succeed, achieving their result, or the world state is unchanged.

Given these simplifying assumptions, one can ask "What does planning buy me?" What is the worth of planning to goal satisfaction? Since these autonomous systems are operating in domains characterized by uncertainty, this becomes "What is the expected value of planning?" by definition, the expected value of any given plan can be given as:

$$E(Plan) = P(SuccessfulExecution) * Value(Plan)$$

and the expected value of planning is the sum of the expected values of all plans.

To determine this expected value, we will need expressions for both the probability of the plan succeeding, and an estimate of the value of the plan. Recall that we are working in an interleaved planning/execution model. We cannot assume that the elaboration of a plan implies its successful execution. Furthermore, it may require several planning/execution sequences to achieve the systems goals.

Plan Probability

From assumptions 1 and 2, (above), it is reasonable to model the successful execution of the entire plan as the product of the probabilities of success of each step in the plan. As a result, the probability of successful execution is a monotonically non-increasing function of plan length, and can be modeled as

$$P(SuccessfulExecution) = \prod_{steps} P(StepSuccess)$$

Since the plan steps are presumed to be independent, and all steps must succeed if the entire plan is to execute successfully, the probability of the entire plan executing is the product of the probabilities of the individual steps. Estimating the value of the plan will be less straight forward, and so a mild diversion is in order.

Plan Value

Traditionally, the value of a plan is considered to be a binary value: 1 if the plan achieves the goal, and 0 if it does not. Beyond this, in classical planning, the shortest feasible plan is considered optimal. However, There are other possible measures of plan success, especially in a domain where plan steps fail, and unexpected events occur.

In feedback control models, there is the notion of an error term. In the case of a single variable controller (such as a system that maintains a temperature set-point), the error term is simply the difference between the desired temperature and the actual temperature.

Using a state-space model of planning, the world can be represented as a high dimensional state vector, and plan steps cause transitions from one state to another. In this model, any two distinct world states are separated by some distance and the goal of the autonomous planning and execution system is to reduce this distance to zero. This distance might be calculated as a Hamming distance, an edit distance, or some other measure.

Using such a distance measure allows the system to express partial goal satisfaction, since the value of a plan can be viewed as the amount of reduction of the distance between the resulting world state and the goal state. The value of a plan which reduces the error term to zero can be normalized to 1.0, and the value of a plan which does nothing can be defined as zero. Clearly, at any intermediate point in a plan, the world state might be

further away from the goal than the initial state, resulting in an intermediate negative plan value. However, any plan that has a final negative value would be rejected by the

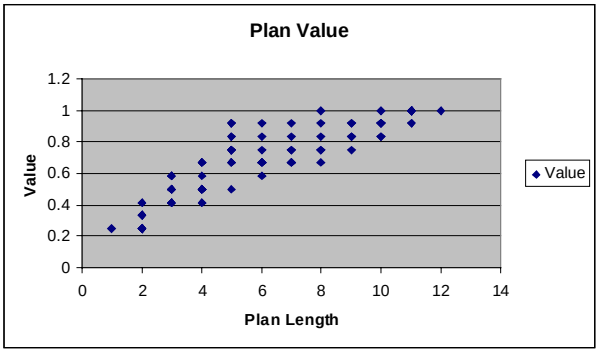

Figure 5 Plan value grouped by plan length

planning system.

These plan values can be classified by the number of steps in the plan, and grouped accordingly (See Figure 5). From this, a mean can be established for the value all plans of length 1, length 2, length 3, and so on. Presumably, shorter plans will be capable of decreasing the error term less than longer plans (simply because the longer plans can apply more operators) and above some

$$Value\ (Plan) \approx 1 - \frac{a}{1 + 2^{length}}$$

length K, all plans will achieve the goal, producing the maximum value. By this argument, we should expect to see a mean Value curve that looks like Figure 6 (below).

This function can be approximated with a logistic curve, where the 'a' term is related to the distance from the initial state to the goal state. This approximation for the value of a plan, is used primarily to express the predicted shape of the plan value curve, and has not yet been experimentally confirmed.

Expected Plan Value

Since the expected value of an event is the product of the probability and the value, we can construct a very rough approximation of the expected value of a plan as a function of plan length (see Figure 7). The relationship between the expected value of a plan and plan length is the result of two aspects in a dynamic tension. First, the idea that it is possible to solve more problems, and possibly produce higher quality solutions with longer plans, suggests that the value of longer plans is higher. Second, the fact that when operating under uncertainty, the value of the later steps in a plan must be discounted, since the probability that all the preceding steps will succeed decreases. The result of these two competing aspects is that to be robust in the face of uncertainty, plans must be long enough to be effective, but not so long that they are unlikely to succeed.

As Figure 7 clearly shows, there is a plan length at which the expected value is a maximum. This length is dependent on three things: the goal, the world state, and the levels of uncertainty in the environment.

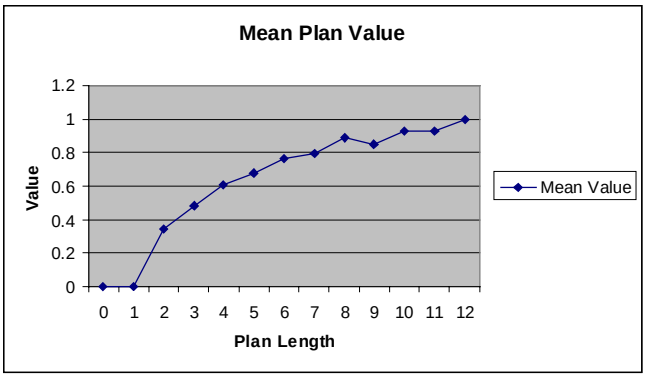

Figure 6 Mean Plan Value as a function of Plan Length

Current Research

Our current research includes embedding effective planning horizon functionality into our existing simulator and probability-aware planning system to empirically validate the effective planning horizon concept. In addition we will investigate the relationship between probability aware planning systems and the effective planning horizon concept, by using our existing planners in comparison with traditional planning systems.

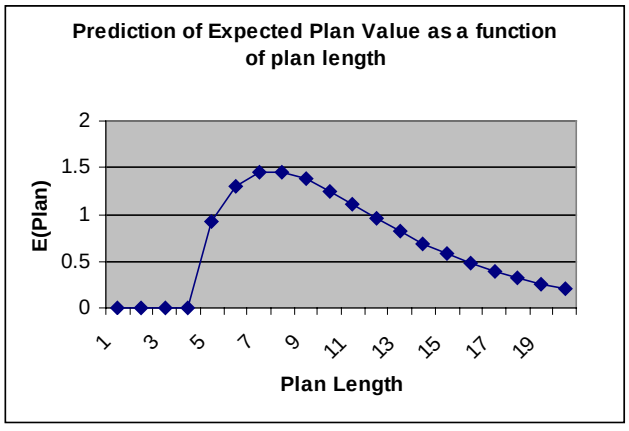

Figure 7 Predicted relationship between plan length and expected value

Conclusions

It is generally accepted that making long, detailed plans in uncertain environments is not effective. In economics, it is common to 'discount' the value of future investments, because the uncertainty in the marketplace reduces the likelihood that they will pay off. For an autonomous system, the question is how to estimate the value of planning, and when to limit deliberation.

Using an underlying, common framework loosely based on feedback control, we have presented a structure for the comparison of common robotic architectures, and guidelines for controlling the allocation of resources to deliberation and execution in an interleaved planning/execution system.

This framework allows the extraction of a control parameter – plan length, which can be adapted to respond to changes in the environment. This ability to autonomously adapt to the environment promises to make autonomous systems more robust in the face of changing environments, and to increase their ability to achieve goals.

Ongoing work to quantify the relationship between environmental uncertainty and the effective planning horizon will result in guidelines to allow situated autonomous systems to adapt to their environments as those environments change.

Acknowledgements

This work was partially supported by funding from the Medical Automation Research Center of the University of Virginia.

References

Bratman, M. E., Israel D. J. and Pollack M. E. 1988. Plans and Resource-Bounded Practical Reasoning. *Computational Intelligence*, 4(4):349 -355

Brooks, R. A. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* RA-2(1):14 -23

Fikes, R. E., Hart, P. E., and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2(3/4):251 – 288

Franklin, G. F., Powell, J. D., and Emami-Naeini, A. *Feedback Control of Dynamic Systems 3rd ed.* Chapt. 3. Addison-Wesley 1994

Gunderson 2000. Adaptive Goal Prioritization by Agents in Dynamic Domains. *IEEE Conference on Systems, Man, and Cybernetics 2000,* pgs 1944 - 1949

Gunderson, J. P. and Martin, W. N. 2000. The Effects of Uncertainty on Plan Success in a Simulated Maintenance Robot Domain. *Journal of Experimental and Theoretic Artificial Intelligence* 12(2000): 153 – 164

Kaga, T., Starke, J., Molnar, P. Schanz, M. and Fukuda, T. 2000. Dynamic Robot-Target Assignment – Dependence of Recovering from Breakdowns on the Speed of the Selection Process. In Parker, L. E. Bekey, G. and Barhen, J. (eds.) *Distributed Autonomous Robotics Systems 4*, Tokyo.:Springer-Verlag

Newell, A. and Simon, H. A. 1963. GPS: A Program that Simulates Human Thought. In Feigenbaum, E. A. and Feldman, J. eds. *Computers and Thought* New York, NY.: McGraw-Hill.

Simmons, R. G. 1994. Structured Control for Autonomous Robots. *IEEE Transactions on Robotics and Automation* 10(1): 34 -43

Walter, W. G. 1950. An Imitation of Life. *Scientific American* May 1950: 42 -45