An Analysis of Sensor-Based Task Planning

Duane Olawsky
Secure Computing Corporation
2675 Long Lake Rd.
Roseville, MN 55113
(612) 628-2846
olawsky@sctc.com

Kurt Krebsbach
Honeywell Technology Center
Automated Reasoning Group
3660 Technology Drive
Minneapolis, MN 55418
(612) 951-7352
krebsbac@src.honeywell.com

Maria Gini
Dept. of Computer Science
University of Minnesota
200 Union Street SE
Minneapolis, MN 55455
(612) 625-5582
gini@cs.umn.edu

Abstract

We present a planner which can plan to perform sensor operations to allow an agent to gather the information necessary to complete planning and achieve its goals in the face of missing or uncertain environmental information.

We have identified and addressed two of the chief problems associated with interleaving execution with planning. The first is that early execution of actions may interfere with, or even preclude the achievement of goals not yet considered. The second is that data needed for planning may be incorrect or uncertain even after they are sensed from the environment.

We cast the problem as one of choosing between various sensing policies. For each piece of unknown or uncertain information, the planner must decide on the best way to obtain it in order to minimize the execution cost or provide itself with the greatest chance of success overall.

We provide an analysis of the factors that influence plan quality, and suggest methods which use these factors to evaluate and rationally choose among various sensing policies.

Contents

1 Introduction

Traditional approaches to task planning assume that the planner has access to all of the world information needed to develop a complete, correct plan—a plan which can then be executed in its entirety by a robot [Hendler *et al.*, 1990, McDermott, 1992]. Unfortunately, this information about the world may not always be available at plan time. This is particularly true when we consider autonomous robots that must operate under general goals over extended periods in unpredictable and changing environments. When crucial information is missing at plan time, it may be impossible to find a complete plan without obtaining additional information. Fortunately, this information is often available at execution time through the use of sensors. The problem then, is how to incorporate sensory data collected at execution time into the planning process which, in traditional approaches, is completed before execution begins.

We have implemented a planner BUMP (Basic University of Minnesota Planner), which is capable of interleaving planning and execution [Olawsky and Gini, 1990]. BUMP is able to defer portions of the planning process which depend on unknown or uncertain information until the information in question can be obtained through sensors. In such cases, BUMP inserts sensor operations directly into the plan which the agent then executes to enable further planning. Alternatively, BUMP may choose to assume a default value for the uncertain information rather than plan to sense it. We call this distinction the sense/default question, and it has played a central role in guiding our recent research efforts.

Deferral and defaulting each have strengths and weaknesses. Deferral can be attractive with accurate sensors because it reduces environmental uncertainty. However, sensing can become prohibitively expensive. In addition, satisfying preconditions for sensor operations can in itself be time-consuming, and as we will see, increases the probability of performing premature actions. Defaulting, while usually less reliable, allows the planner to complete more of the plan before execution begins. This in turn allows the planner to see further into the plan and detect problems which may lie beyond the horizon of the deferral point. As default uncertainty increases however, further planning based on defaults becomes increasingly arbitrary.

In general, it is nontrivial to decide whether to defer planning and sense a given uncertain value or simply choose a default value and continue planning with it (facing the risk of having made an incorrect choice). Deciding on the best strategy for a given planning problem consists of computing the tradeoffs of various strategies, and that is the major topic of this paper.

In Section 2 we outline the results of our previous work in this area describing the major issues involved and the problem domain in which we have worked. Sections 3 and 4 then analyze two different methods for deciding when to sense and when to use a default.

2 Background

The eventual goal of this line of research is to develop improved task planning strategies for increasingly realistic problem domains. In particular, we focus our attention on domains in which some amount of domain information is not certain at plan time, and must be sensed or defaulted before planning can proceed. We first discuss two measures of the quality of a plan. It is useful to think of the quality measures as functions to minimize or maximize, and the strategies as means to that end. Then we provide a summary of factors that influence the quality including three types of planning strategies. After that, we introduce the Tool Box World [Olawsky and Gini, 1990] and discuss the problems posed by it for a planner.

2.1 Plan Quality Criteria

Before proceeding, we should be clear as to the objectives the above-mentioned planning strategies are intended to serve. We refer to these objectives as *plan quality criteria*. In this paper, we limit our attention to just two criteria, namely *execution cost* and *success rates*.

Execution Cost: This criterion measures the cost of all actions to be performed during all phases of execution when the planner is allowed to recover from premature actions (i.e., undo and redo these actions). This provides us with some indication of *how* inefficient the inferior solutions are for those problems judged unsuccessful under the success criterion. In this paper we treat each instantiated process (i.e., action) in the plan as having unit cost, although it would be trivial to assign varying costs to various types of action.[1] Measuring execution cost implies *reversible actions*. For our cost functions to be well defined, we must assume the agent can recover from any such situation at some finite cost.

Success Rates: Success rates reflect the *percentage of problems* in which BUMP is able to construct a plan in which no processes need to be undone as a result of being executed prematurely. One of the major difficulties in interleaving planning and execution is to keep the robot from performing actions which may interfere with goals not yet considered. In our experiments this occurred, for instance, when the robot bolted closed a tool box only to discover that it contained

[1]In [Krebsbach *et al.*, 1991] we also considered planning cost. We will not consider that cost in this paper since it appears to closely parallel execution cost in our domain.

a wrench (or bolt) needed to accomplish a later goal. Under this criterion we consider such plans failures, in effect assuming the agent is unable to recover from such premature action. When we use success rates as the plan quality criterion, we consider any plan which results in premature action as a complete failure. Succinctly stated, measuring success rates implies *irreversible actions.*

2.2 Planning Factors

We have identified the following as important factors of task planning with sensors. Our previous studies have shown that by intelligently controlling these factors, a planner can improve the quality of its plans, often dramatically. We define the term *overall planning strategy* to refer to a set of algorithms for determining each of the following parameters for any given problem instance. This overall strategy will be selected before planning starts. Complete results of the relevant experiments can be found in [Krebsbach *et al.*, 1991].

Goal ordering: The initial goal ordering describes the order in which BUMP will attempt to construct portions of the plan to satisfy each goal. This ordering is fixed when planning commences and does not change. It is important to note however, that this order is not necessarily the order of execution. BUMP is fairly good at ordering and reordering actions to exploit helpful goal interactions and avoid harmful ones. However, execution of a partial plan to obtain sensory data hampers BUMP's ability to reorder actions. Once an action is executed, it can no longer be moved around in the plan. Choosing an initial goal ordering to facilitate intelligent action reordering is one way to improve BUMP's performance. We found it advantageous to order the planner's initial goals based on the amount and type of unknown information at the start of planning. The guiding principle is that most sensing should come as early as possible in the plan. The disadvantage of potentially premature action caused by early sensing is, in most cases, outweighed by the advantage of constructing most of the plan with more information. This goal ordering heuristic depends critically on the assumption that the planner can identify connections between its top-level goals and the unknown domain propositions in the problem. In our experiments there is a one-to-one correspondence between goals and potential unknowns, so the issue is not addressed here. For the remainder of this paper then, we will assume that regardless of the input order of the goals, it is a trivial matter to reorder goals in this way.

When to sense: A critical decision when interleaving planning and execution is *when* to switch from one to the other. In related research, [Olawsky and Gini, 1990] identified two general strategies to manage the transfer of control between the planning and execution modules (i.e., *control strategies*). In both strategies, if the planner discovers that it requires unknown information, it inserts a sensor operation into the plan to obtain the information. It then plans to satisfy any preconditions of this sensor operation.

In the first strategy, known as Stop and Execute (SE), when the planner encounters a goal whose achievement *depends on* information it has planned to sense, control is immediately transferred to the execution module. The sensor process and all processes ordered before it in the current partial plan are executed. Control then returns to the planner.

In the second strategy, Continue Elsewhere (CE), goals whose achievement depends on information that BUMP has planned to sense are deferred. Planning continues elsewhere for other goals. Only when all goals are either planned to completion or deferred is control transferred to the execution module. Execution halts after each sensor operation to allow completion of planning for a deferred goal. In general, CE allows much more planning, albeit less informed planning, to occur ahead of the first execution phase.

These two strategies are the only truly *domain independent* control strategies that we have found useful. It is certainly possible to devise domain dependent strategies for these types of problems. For example, in [Olawsky and Gini, 1990] we described a modified Continue Elsewhere strategy call Sense Before Closing (SBC). This strategy was designed for the Tool

Box World (see the next section), and it attempts to sense all the unknown information before any tool boxes are closed. It would do this by attempting to order all sensor actions before all box closing actions in the plan. SBC could not be directly applied to other domains. Domain dependent strategies tend to be rather brittle, breaking down with only the slightest changes to the problem domain (see Section 3.6.2 for discussion of this). For this reason we decided to begin our study with the domain independent strategies, and that is the subject of this paper. By starting with domain independent strategies we can determine whether they work sufficiently well for any class of problems or whether we need a new domain dependent strategy for each domain. The results presented here demonstrate that it is feasible to use domain independent techniques when the domain does not include irreversible actions. We also show that for domains with irreversible actions the domain independent strategies are clearly not sufficient and we need to develop special-purpose domain dependent strategies such as SBC.

In the remainder of this paper we will assume the planner is employing the domain independent Stop and Execute (SE) strategy, since the empirical data collected in [Krebsbach *et al.*, 1991] suggests that its performance is at least as good as that of Continue Elsewhere, the other domain-independent strategy. The behavior of SE also lends itself more naturally to mathematical analysis.

What to sense: Finally, there is the question of *which* unknown or uncertain quantities to sense and which to default. We call a method for answering this question a *sensing policy.* Choosing the best sensing policy requires careful consideration of domain-specific factors such as default reliabilities, sensor reliabilities, planning costs, execution costs, and the cost of human intervention. Most of this article focuses on choosing a sensing policy.

In summary, an overall planning strategy consists of a goal-ordering heuristic, a control strategy and a sensing policy.

2.3 The Tool Box World

The problem domain for which empirical data was collected in our previous work and which is analyzed in this paper is called the *Tool Box World.* We chose this domain because, although an abstraction of a real-world domain, it contains all of the important elements of interest to us, and lends itself well to systematic analysis.

In the Tool Box World, an agent (i.e., robot) is in a room with n tool boxes $T_1, T_2, \ldots, T_n$, each containing wrenches and bolts of various sizes. The robot knows the initial locations of the wrenches and bolts. Bolts are identified by a unique name, and wrenches are identified by a unique size. The robot has been instructed to close and bolt one or more tool boxes with particular bolts. Each of these bolts has an associated size. To perform each bolting operation, the robot must use the wrench whose size matches the bolt. The bolts are initially in their respective boxes (e.g., bolt b_i is in box T_i).[2] All of the tool boxes are initially open. We assume the robot begins at a neutral site (one unrelated to any work that it must do). Since the planner's goals are strongly associated with particular tool boxes, this assumption was meant to avoid any bias. A sensor is available that can classify bolts by their head size (e.g., a number from 1 to 10). For simplicity, the bolt sizes are indicated along the same scale as the wrench sizes. We also assume the robot has a tool belt into which it can put an unlimited number of bolts and wrenches.[3]

Given that we assume prior goal ordering and the SE control strategy, the following are the parameters which describe the problem space:

[2]This causes the robot to see less of the world while solving its early goals since it need not go anywhere to get a bolt. While this may at first appear to simplify the problem, in effect it tests the planner on a slightly more difficult set of problems than it would encounter by chance. The more places BUMP visits, the more of a chance it has to gather other information, quite possibly information it could use to make more informed action ordering decisions. This in turn would decrease BUMP's vulnerability to failures due to premature action.

[3]We are not concerned here with the arm-empty conditions as used in typical definitions of the blocks world. Our main goal in defining this domain is to study how sensor use can be interleaved with planning.

Number of Tool Boxes (b) The number b also denotes the number of initial goals, since we assume there is exactly one goal associated with each tool box. All of our experiments assumed b wrenches, but this assumption is relaxed somewhat in Section 3.

Number of Unknowns (u) The number u denotes the number of unknown bolt sizes at the start of planning. This value is subject to the constraint $u \leq b$. When a default is used, we will consider the bolt size to be *known* at the start of planning since BUMP need not defer any goals that depend upon this default information. Thus, u can also be considered the number of bolt sizes that will be determined by a sensor reading.

Initial Wrench Locations: Each of the wrenches may initially be in any tool box, implying b^b possible wrench placement scenarios (assuming b wrenches.

2.4 Types of Failure

In the mathematical characterization of the Tool Box World described later in Sections 3 and 4 we take into account two very different ways in which a plan can fail. The first of these we call failure due to *premature action*. Premature action occurs when the agent executes an action A before it realizes a higher quality plan would result from delaying the execution of A until after the execution of other actions. This is probably the most significant difficulty with interleaving planning and execution. Any actions performed before planning is completed are potentially premature.

A second type of failure is due to *bad data*. Bad data can be in the form of an incorrect default or sensor reading. As is the case for premature action, bad data implies total failure in a success-based analysis, but means increased cost (for recovery) in a cost-based analysis.

The analysis and algorithms presented in the remainder of the paper are intended to improve task planning for a general class of problems. They either minimize *expected cost* or maximize *expected success rates*. Much of the ensuing analysis depends either on aggregate experimental data obtained in our earlier work, or on probabilistic models of the domain. This is necessary because the methods we propose are to be employed prior to any planning or execution, so the outcome of a particular sensor process cannot be known for certain, only probabilistically modeled.

2.5 General Characteristics of the Domain

Before presenting an in-depth analysis of the Tool Box World in Sections 3 and 4, we briefly discuss the general characteristics of the Tool Box World that influence our analysis. This provides some idea of the range of problems to which our ideas could reasonably be applied. We will comment further in Section 3.6.3 on applying these ideas to other problem domains.

The most important characteristic is that there is information missing at planning time and the planner can request this information by including sensor actions in the plan. This implies some interleaving of planning and execution. This missing information in itself produces no difficulties unless it can cause the execution of an action that makes it impossible to carry out a later action or makes the later action unnecessarily expensive.

Premature actions are likely to occur in any domain where there is detrimental interaction between goals. A planner with sufficient knowledge of the world can often avoid these premature actions by reworking the plan prior to any execution. However, as we shall see in Sections 3 and 4, when working with incomplete information, the planner's ability to avoid premature actions declines precipitously.

Another domain characteristic that we have studied is the effect of bad sensor data on the performance of the planner and robot. This characteristic is present in virtually all domains that assume the use of sensors.

A final domain characteristic assumed in this work is a strong relationship between the top-level problem description and the missing pieces of information. Our techniques for using the analysis to improve system performance focus on selecting a sensing policy (e.g., default or sense) for each piece of unknown information. These policies are chosen prior to all planning. Thus, we must be able to identify the unknown information by examining only the problem description (i.e., the top-level goals and initial state information). In the Tool Box World there is at most one unknown per top level goal and this has facilitated our analysis. We hypothesize that a similar analysis could be done in domains with multiple unknowns corresponding to each top-level goal as long as there is no detrimental interaction between the unknowns for a single top-level goal, but this is beyond the scope of the current research.

We consider two versions of the Tool Box World in the remainder of this paper — one that assumes reversible actions (the cost-based analysis in Section 3) and another that assumes irreversible actions (the success-based analysis in Section 4). Both versions are amenable to the type of analysis done in this paper, and the analysis allows a deeper understanding of the domain. For example, our success-based analysis demonstrates the extreme difficulty of planning with incomplete information using irreversible actions.

3 A Cost-Based Analysis

When the robot is able to detect at some point that a default value or a sensor reading was erroneous and then take corrective actions, it makes sense to use cost as the quality criterion. In this section we explore the effect of unknown information on execution cost. A similar analysis could be developed for planning effort. While the algorithms we present make use of a probabilistic analysis of the Tool Box World, it should be noted that they could also be applied in other domains using approximations obtained empirically.

3.1 Base Costs

The approaches presented in this paper focus on selecting a sensing policy—that is, deciding when to use a sensor and when to use a default. When recovery is possible, we not only make a selection of which option to try but also what to do if that option gives us incorrect information. For example, if a decision is made to try a default which later turns out to be incorrect, BUMP could try again by using a sensor to obtain the information. If the sensor reading also fails, it might be possible to obtain the required information through human intervention (presumably at a very high cost).[4] We will use the letters D, S and I to describe these three options—default, sense and intervene. The letters will be concatenated in various combinations to describe different *event sequences.* For example, the sequence SDI describes the scenario where an incorrect sensor reading is followed by an incorrect default value which is followed by successful human intervention. It is assumed that the last resource is always successful. DS describes the event sequence where an incorrect default value is tried followed by a correct sensor reading. An entire problem can contain multiple unknowns, and for each of these there will be an event sequence.

The event sequences that actually occur depend upon the correctness of the sensor and default readings which is not known prior to execution. Thus, we probabilistically analyze the expected cost under various scenarios. It will make our analysis simpler to break this cost into two components:

Recovering from Bad Data - The cost of recovering from errors due to bad sensor readings or incorrect default values. This cost will be analyzed in Section 3.4.

All Other Costs - This includes the cost of actions needed to solve the problem when everything works perfectly plus the cost of recovery from premature actions. This combined cost will be denoted by $C'(b, u, h)$ where b is the number of boxes, u is the number of unknowns and h is as defined below.

[4]Some other options that we do not consider in this analysis are

1. to try a different sensor, or
2. to continue trying the same sensor.

If the sensor is working at all (i.e., there is a non-zero probability of a correct reading), then with persistence the second option should eventually produce a correct reading. The probability of n readings all being incorrect goes to 0 as $n \to \infty$. This might also have a very high cost. Furthermore, it might be difficult for the planner to know whether the sensor is functioning at all. If not, then the planner is wasting its time using the sensor repeatedly.

The cost function C' can be written as the sum of several component costs:

$$C'(b,u,h) = b \cdot \beta + u \cdot \sigma + h \cdot \eta + \rho(b,u) \cdot \pi \tag{1}$$

where

$\beta =$ The base cost of simply achieving each goal, assuming no sensing and no recovery.

$\sigma =$ The cost of executing a single sensor process.

$\eta =$ Cost of a human intervention (i.e., using a human as a sensor).

$\pi =$ The cost of recovering from a single premature action.

$h =$ The number of decision points with event sequence I. (There is one decision point for each unknown.)

$\rho(b,u) =$ The expected number of premature actions (and therefore, recoveries) with b boxes and u unknowns.

Several of these values can be approximated rather easily by examining the empirical data and thinking about the actions that take place in various circumstances. The base cost of achieving a goal can be anywhere from six actions (goto box, pickup bolt, pickup wrench, close box, insert bolt and tighten bolt) to eight (goto box, pickup bolt, goto box, pickup wrench, goto box, close box, insert bolt and tighten bolt). Empirically, the shorter action sequences are more common, so, assuming a unit cost per action, we approximate

$$\beta = 6.5.$$

A sensor process only requires one action in addition to the base actions, but sometimes an extra goto is inserted (to get the correct wrench) which could have been avoided if BUMP had known the size of the bolt ahead of time. So, we take

$$\sigma = 1.5.$$

Finally,

$$\pi = 7$$

since it always takes seven additional steps to recover from a premature action (loosen bolt, remove bolt, open box, get wrench, close box, insert bolt and tighten bolt). The value of η is set as a system parameter. In the next section we will obtain a formula for $\rho(b,u)$.

Implicit in formula (1) is the following simplification of the problem: we ignore the effect that bad data has on premature actions. In general, we expect bad data to *decrease* premature action slightly since the recovery from bad data will cause the robot to pick up additional wrenches. Having more wrenches decreases the chance of premature action.

3.2 Expected Number of Premature Actions over Problem Space

In modeling cost, we need a way of computing the total number of premature actions over an entire problem set. When there are multiple premature actions for a single problem, we must count them all since there will be a separate cost in recovering from each of them. We will compute this by finding for each wrench the number of problems where that wrench will be bolted into a tool box prematurely. We will then sum these values.

We first characterize the conditions which cause BUMP to perform a premature action in the Tool Box World. Using this, we can calculate the expected number of premature actions with b boxes and u unknowns. Throughout our analysis we assume the top level goals are reordered, as described in Section 2, so as to minimize the average cost. In addition, we focus on the Stop and Execute control strategy. As a starting point we consider two examples involving 3 boxes and 2 unknowns. After that we consider cases where there are b boxes and exactly 2 unknowns. Finally, we generalize this result for b boxes and u unknowns.

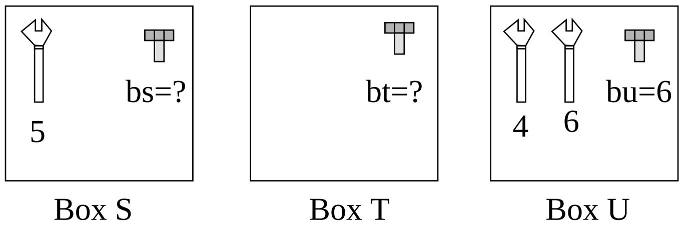

Figure 1: Initial State: Problem 1.

3.2.1 Two Examples

Let us begin with a specific case involving 3 tool boxes S, T and U that are to be bolted closed with bolts b_s, b_t and b_u, respectively. The sizes of these three bolts are 4, 5 and 6, respectively, but only the size of b_u is known at the start of planning. Furthermore, we assume that the goals are considered by the planner in the order S, T, then U. As mentioned earlier, we also assume each bolt is in the box for which it is used (i.e., b_s is in S, b_t is in T and b_u is in U).

Example 1. There are 27 problems determined by the possible initial locations of the three wrenches 4, 5, and 6. To clarify the following discussion, we first outline the progress of the system on one of the problems where the system performs a premature action: wrench $4 \in U$, wrench $5 \in S$, and wrench $6 \in U$ (Figure 1). The first planning phase begins with the consideration of the goal (Bolted S b_s). Since the size of b_s is unknown, BUMP first plans to sense b_s. It cannot, under Stop and Execute, plan further until it obtains the actual size. So, execution begins. The robot goes to box S, grabs the bolt b_s and senses it (obtaining $b_s = 4$). Planning resumes, and BUMP plans to go to box U, get wrench 4, come back to S and bolt it closed. It then starts planning to achieve the goal (Bolted T b_t). Again, it plans to sense a bolt (this time b_t). Since this involves going to box T to get the bolt, the sense operation is ordered after the completion of the S goal.[5] At this point BUMP again stops to obtain sensor information. All actions ordered before the sensor operation must be executed first. Thus, the robot goes to U, gets wrench 4, goes back to S, bolts S shut, goes to T, picks up b_t and senses it (with result $b_t = 5$). Now planning continues. BUMP plans to get wrench 5, but at this point it realizes that a premature action has been committed (i.e., bolting S shut with wrench 5 in it). The robot must then recover from the premature action by opening S, getting wrench 5 and rebolting S. □

Obviously, a special-purpose strategy such as Sense Before Closing could avoid the premature action and do better on this problem. However, a different strategy would have to be developed for each domain, and this could be difficult for domains that are more complex than the Tool Box World.

Example 2. Let us briefly consider a similar problem (Figure 2): wrench $4 \in T$, wrench $5 \in S$, and wrench $6 \in S$. The initial planning phase is identical. However, during the second phase, BUMP plans to go to T instead of U to get wrench 4. When it plans to sense bolt b_t, it decides to perform this operation while it is at T getting the wrench. This sensor operation is therefore performed (and planning resumes) before box S is closed. At that point, the rest of the plan is completed since

[5]This follows the reasonable convention of finishing up what you are doing at one location before going on to another.

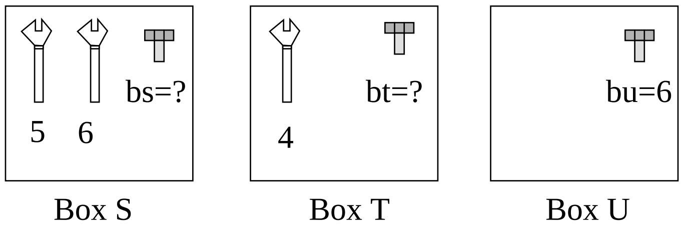

Figure 2: Initial State: Problem 2.

no more sensor readings are needed and planning need not be suspended. BUMP now has complete information and will plan to get wrenches 5 and 6 out of S before closing it. □

These two examples demonstrate the behavior we must describe. There are two possible premature actions in Example 2. One is to prematurely lock wrench 5 in box S, and the other is to lock wrench 6 in S. In both cases the premature action must occur before bolt b_t is sensed since once this sensor operation occurs the planner has all necessary information and a complete plan is found. Any problems can still be remedied (by action reordering) in the final planning phase. Thus, BUMP can avoid premature actions if something causes it to sense b_t before closing S. This will happen whenever the robot must travel to box T (the location of b_t) before closing S, and this in turn will happen if wrench 4 is in T. Thus, wrench 5 can be locked in box S only if wrench 4 is not in T. Wrench 6 could be in any box, so this gives six problems that involve a premature action affecting wrench 5. Similarly, wrench 6 can be locked in box S only if wrench 4 is not in T. This gives six problems involving a premature action affecting wrench 6.

3.2.2 Finding $\rho(b, 2)$

Now let us move to the case where there are an arbitrary number b of boxes and exactly 2 unknowns. We first introduce some notation. Assume as before that the goals are reordered so that all the ones with unknown information come first, followed by those for which all information is known. Let $g_1, g_2, \ldots, g_b$ be these reordered goals. Now, define $W(g_i)$ to be the wrench needed for goal g_i and $B(g_i)$ to be the box associated with that goal. Since $u = 2$, the sizes of the bolts in $B(g_1)$ and $B(g_2)$ are unknown. The only type of premature action that can occur is for one or more of the wrenches $W(g_2), W(g_3), \ldots, W(g_b)$ to be locked in $B(g_1)$. If $W(g_1)$ is in $B(g_2)$ then no premature actions will occur because a complete plan will be found before any boxes are closed. If $W(g_1)$ is not in $B(g_2)$ and some wrench $W(g_i), i \geq 2$, is in $B(g_1)$ a premature action involving $W(g_i)$ will occur no matter where the other wrenches are located. This gives $(b-1)b^{b-2}$ potential premature actions for each of the wrenches $W(g_2), W(g_3), \ldots, W(g_b)$ for a total of $(b-1)^2 b^{b-2}$.

3.2.3 Finding $\rho(b, u)$

Now, we consider the general problem with b boxes and u unknowns. There are two ways that the placement of a wrench can lead to a premature action.

1. $W(g_i) \in B(g_j)$ where $j \leq i - 2$ and $j < u$. Informally, a wrench needed for a given box is in the box for a goal that is two or more positions back in the sorted goal list. At the time when the robot is closing $B(g_j)$, BUMP will not know yet that $W(g_i)$ is needed. We call this condition *2-back*. We must note however that a *2-back* failure will not occur if $j = u - 1$ and

$W(g_j) \in B(g_u)$ since the final unknown bolt, which is in $B(g_u)$ will be sensed before $B(g_j)$ is closed (consider wrench 6 in Example 2 above).

2. $W(g_i) \in B(g_{i-1}) \wedge W(g_{i-1}) \notin B(g_i)$ where $i \leq u$. Informally, a wrench needed for a given box B is in the preceding box, and the wrench for the preceding box is not in B. If the wrenches were "flipped" (i.e., $W(g_i) \in B(g_{i-1}) \wedge W(g_{i-1}) \in B(g_i)$), the robot would sense the bolt for g_i before closing $B(g_{i-1})$ and the premature action would be avoided. We call this condition *half-flipped.* (Consider wrench 5 in Examples 1 and 2 above.)

Let us begin by considering the wrenches corresponding to bolts of unknown size (i.e., $W(g_1), \ldots,$ $W(g_u)$). Assume that the reordered list of goals is $g_1, \ldots, g_b$ with $g_1, \ldots, g_u$ $(u \leq b)$ having unknown bolt sizes. For each wrench $W(g_i)$, $(i = 3, \ldots, u)$ there exists $i-2$ locations, $B(g_1), \ldots, B(g_{i-2})$ that would cause a *2-back* failure. The remaining $b-1$ wrenches can be in any of the b boxes, so for wrench $W(g_i)$, there are $(i-2)b^{b-1}$ *2-back* failures. Note that there cannot be any 2-back failures involving wrenches $W(g_1)$ and $W(g_2)$. Summing over the wrenches $W(g_3), \ldots, W(g_u)$ gives the total number of *2-back* failures involving unknowns:

$$\begin{aligned} F_1(b,u) &= \sum_{i=3}^{u}(i-2)b^{b-1} \\ &= b^{b-1}\frac{(u-2)(u-1)}{2}. \qquad (2) \end{aligned}$$

Note that formula (2) also happens to be correct for $u = 2$ (giving zero), so it applies to all $u \geq 2$.

Now consider half-flipped failures for the wrenches $W(g_1), \ldots, W(g_u)$. A *half-flipped* failure occurs when $W(g_i) \in B(g_{i-1}) \wedge W(g_{i-1}) \notin B(g_i)$. This allows 1 possible location for $W(g_i)$, $b-1$ locations for $W(g_{i-1})$ and b locations for each of the other $b-2$ wrenches. Thus, for $i = 2, \ldots, u$, there are $(b-1)b^{b-2}$ problems involving a *half-flipped* error on wrench $W(g_i)$. There cannot be any half-flipped failures for $W(g_1)$. The total number of *half-flipped* failures for these wrenches is thus:

$$F_2(b,u) = (u-1)(b-1)b^{b-2}. \qquad (3)$$

Formula (3) is correct for all $u \geq 2$.

Finally, we examine cases where a wrench corresponding to a goal with a known bolt size is bolted into a box. These cases are different because once the final unknown is sensed, planning is completed, and reordering can be done before any further (possibly premature) actions are executed. As a result, there are no half-flipped failures for these wrenches. However, a *2-back* failure can still occur since a known wrench can be locked in an unknown box (other than the final unknown box). If a wrench $W(g_i)$ with $u < i \leq b$ is in a box $B(g_j)$ with $1 \leq j \leq u-2$, there will be a 2-back failure on wrench $W(g_i)$. Since the other wrenches can be anywhere this accounts for $(b-u)(u-2)b^{b-1}$ premature actions. In addition, if $W(g_i)$ is in $B(g_{u-1})$ there will be a 2-back failure if $W(g_{u-1})$ is not in $B(g_u)$. The other wrenches can be anywhere. This gives another $(b-u)(b-1)b^{b-2}$ premature actions. The total number of 2-back failures for wrenches $W(g_{u+1}), \ldots, W(g_b)$ is

$$\begin{aligned} F_3(b,u) &= (b-u)(u-2)b^{b-1} + (b-u)(b-1)b^{b-2} \qquad (4) \\ &= (b-u)(bu-b-1)b^{b-2} \qquad (5) \end{aligned}$$

for $u \geq 2$.

Summing formulas (2)–(5) gives us the following formula which represents the total number of premature action failures for the class of problems involving b boxes and u unknowns.

$$\begin{aligned} F(b,u) &= F_1(b,u) + F_2(b,u) + F_3(b,u) \\ &= b^{b-1}\frac{(u-2)(u-1)}{2} + (u-1)(b-1)b^{b-2} + (b-u)(bu-b-1)b^{b-2} \\ &= b^{b-2}\left[\frac{(u-1)(2b-u)b}{2} - b + 1\right]. \end{aligned}$$

This formula is valid for all $u \geq 2$.

Since b^b is the number of problems in the entire problem space for b tool boxes, the ratio $F(b,u)/b^b$ describes the number of expected failures for a randomly selected problem from that space. This is $\rho(b,u)$ introduced in Section 3.1. With $u \leq 1$ there are no premature actions since a complete plan is found before any boxes are closed. So,

$$\rho(b,u) = \begin{cases} \frac{(u-1)(2b-u)}{2b} + \frac{1-b}{b^2} & \text{if } u \geq 2, \\ 0 & u = 0, 1. \end{cases}$$

It is instructive to hold u fixed and take the limit as $b \rightarrow \infty$.

$$\lim_{b \rightarrow \infty} \rho(b,u) = u - 1.$$

This suggests that it is the absolute amount of unknown information that plays the primary role in determining premature actions, not the number of boxes. Remember that $u - 1$ is the limit of the *expected* number of premature actions. Clearly, for a given problem instance it is possible that the number of premature actions is greater than 1.

3.3 Expected Cost of a Policy

In this section we describe the formula for the expected cost of a policy. It will be based upon the two components of cost given above: recovery from bad data, and $C'(b,u,h)$.

Bad data can be either a sensor reading or a default value, and the recovery can use a sensor, a default or human intervention. Since default values are merely guesses about the state of the real world they are obviously fallible. A bad sensor reading could be due either to a malfunction of the sensor or simply to noise in the sensor data. We define the *reliability* of a value to be the probability that it is correct. Reliability is a measure of a priori confidence that the value is correct. No notion of amount of error or distance from the correct value is considered here. This definition of reliability applies equally well to sensor and to default values.

The expected cost of recovering due to bad data will be denoted by a function $\delta_{b,w}$ where b is the number of boxes, and w is the number of wrenches. This function takes b arguments each describing the event sequence for a single decision point. Each argument is a sequence of the letters D, S and I as described in Section 3.1. Thus, $\delta_{2,3}(DSI, DS)$ is the cost of recovering from bad data when there are two boxes and three wrenches, and the event sequences for the two unknowns are DSI and DS. The order of the arguments represents the order in which the decision points are reached during execution.

Given these definitions we can develop formulas for the expected cost of a sense/default/intervene policy. We introduce another function $K_{b,w}$ to denote this expected cost for b decision points and w wrenches. The function $K_{b,w}$ takes b arguments which describe the policy used. The arguments have the same form as the arguments of $\delta_{b,w}$, but the interpretation differs slightly. Whereas the event sequence SDI implies that the sensor and the default both yielded bad information, this is untrue for the policy SDI. The policy represents a decision about which resources will be *tried if necessary.* That is, SDI denotes the following policy:

1. Use a sensor.
2. If the sensor gives incorrect information, try a default.
3. If the default is incorrect, request intervention.

The resulting event sequence could be S, SD, or SDI. In contrast to an event sequence, each component (e.g., DSI) of a policy must end in I so that the robot can always succeed and a finite cost can be assigned (we assume intervention always yields correct information). The function $K_{2,3}(DSI, DI)$ is the expected cost with two boxes and three wrenches using the policy DSI-DI and taking into account the likelihood that the defaults and sensors will yield correct information.

$K_{b,w}$ can be defined as a weighted sum of C' and $\delta_{b,w}$ values. The weights reflect the likelihood of particular event sequences based upon the reliabilities of the sensor readings and default values.

We will use the notation r_i to denote of the reliability of a default value for decision point i and s_i to denote the reliability of the corresponding sensor reading. As an example, with one decision point and w wrenches the expected cost of the policy SDI is expressed by the weighted sum:

$$K_{1,w}(SDI) = C'(1,1,0) + (1-s_1)r_1\delta_{1,w}(SD) + (1-s_1)(1-r_1)\delta_{1,w}(SDI).$$

The first term represents the base cost required as a minimum to achieve the goal assuming no bad data is encountered. The second term represents additional costs incurred when the sensor reading is bad but the default works. The final term reflect the additional cost when both the sensor and the default are bad and human intervention is required. The arguments of C' are determined from the total policy. The value of u will be the number of decision point policies that begin with S, and the value of h will be the number of them that begin (and end) with I.

Using similar reasoning, we obtain the following formula for the policy DSI, in which we try the default before the sensor:

$$K_{1,w}(DSI) = C'(1,0,0) + (1-r_1)s_1\delta_{1,w}(DS) + (1-r_1)(1-s_1)\delta_{1,w}(DSI).$$

There are three other possible policies with one decision point which are described by the following formulas:

$$\begin{aligned} K_{1,w}(SI) &= C'(1,1,0) + (1-s_1)\delta_{1,w}(SI) \\ K_{1,w}(DI) &= C'(1,0,0) + (1-r_1)\delta_{1,w}(DI) \\ K_{1,w}(I) &= C'(1,0,1). \end{aligned}$$

Given a way to calculate $\delta_{1,w}$ we could select a policy with probabilistically optimal performance by calculating the value of all five of these formulas and taking the policy with the minimum value. However, this algorithm will not work in general because it is far too expensive. To see this we look at one of the formulas for two decision points. One policy would try both defaults first, backed up by sensing and intervention. The resulting weighted sum is

$$\begin{aligned} K_{2,w}(DSI,DSI) = \; & C'(2,0,0) + (1-r_1)s_1r_2\delta_{2,w}(DS,D) \\ & + r_1(1-r_2)s_2\delta_{2,w}(D,DS) + (1-r_1)s_1(1-r_2)s_2\delta_{2,w}(DS,DS) \\ & + (1-r_1)(1-s_1)r_2\delta_{2,w}(DSI,D) + r_1(1-r_2)(1-s_2)\delta_{2,w}(D,DSI) \\ & + (1-r_1)(1-s_1)(1-r_2)s_2\delta_{2,w}(DSI,DS) \\ & + (1-r_1)s_1(1-r_2)(1-s_2)\delta_{2,w}(DS,DSI) \\ & + (1-r_1)(1-s_1)(1-r_2)(1-s_2)\delta_{2,w}(DSI,DSI). \end{aligned}$$

This formula is certainly much more complicated than the formulas for one decision point. In fact the number of terms to be summed in a formula that considers all three resources — defaults, sensors and intervention — grows exponentially (3^b for b decision points). The number of factors in the longest term is $2b+1$. Calculating the expected cost of just one policy is $O(b3^b)$, and the number of policies to be evaluated and compared is 5^b. Clearly, we cannot use this exhaustive method to find the optimal policy unless the number of decision points is quite small.

In the next section we will consider the calculation of $\delta_{b,w}$, and in Section 3.5 we will present an algorithm to solve a slightly simplified version of the policy selection problem.

3.4 Cost of Recovery from Bad Data

Unlike the cost of recovering from a premature action (which is a constant seven steps), the cost of recovering from bad data varies dramatically with the context in which the bad data is encountered. This context can be broken down into three components: the current state of the world, the method of obtaining new information to be employed in the recovery, and the information that is obtained. Unfortunately, the state of the world (in particular, the set of wrenches in the robot's tool belt) depends on the sensor readings that have occurred. Since this information is not known until

Location of Wrench	Recovery Action Sequence	Recovery Cost
in tool belt	tighten	1
in an open box	goto, pickup wrench, goto, tighten	4
in the closed box robot is trying to bolt	remove bolt, open, get wrench, close, insert bolt, tighten	6
in some other, bolted box	goto, loosen, remove bolt, open, get wrench, close, insert bolt, tighten, goto, tighten	10

Table 1: Using a default value to recover from bad data.

Location of Wrench	Recovery Action Sequence	Recovery Cost
in tool belt	remove bolt, sense, insert bolt, tighten	4
in an open box	remove bolt, sense, insert bolt, goto, pickup wrench, goto, tighten	7
in the closed box robot is trying to bolt	remove bolt, sense, open, get wrench, close, insert bolt, tighten	7
in some other, bolted box	remove bolt, sense, insert bolt, goto, loosen, remove bolt, open, get wrench, close, insert bolt, tighten, goto, tighten	13

Table 2: Using a sensor to recover from bad data.

execution time, it is impossible to know before planning begins the exact context during any arbitrary recovery. Nevertheless, we can learn a good deal by characterizing the effects of the context on recovery, and that is what we do in this section.

Tables 1 and 2 describe the costs in the various contexts in which recovery from bad data can occur. Each row is a world state context, and each table assumes a given recovery method. For example, consider the case where we recover from bad sensor data by using a default value (see Table 1). The bad data will be discovered when the robot attempts to tighten the bolt using an incorrect size of wrench. The default indicates a wrench to try next, and the first column specifies the location of this wrench in the current state. If that wrench is in the box the robot is currently attempting to bolt, the third row of the table applies.

It is in principle impossible to know prior to executing a portion of the plan exactly which of the eight recovery sequences will be used. However, using these context-dependent recovery costs, we can determine the expected cost of recovering from bad data by weighting each cost from the tables by the likelihood of the agent being in the associated context. This likelihood can be determined based upon four pieces of information. In addition to b and w, we also need the following information to model the expected cost:

$h =$	The number of wrenches the agent is currently holding (including those in its tool belt).
$c =$	The number of boxes currently closed.

The values of b and w remain constant for a given problem. Since each goal involves bolting closed a box, c reflects the number of goals achieved so far.

Suppose we want to know the expected cost of recovery from bad data at an arbitrary decision point assuming we try sensing first, it fails, and then defaulting works (event sequence SD). At the time when the sensor fails and we are ready to try a default, we could potentially be in any of the contexts listed in Table 1. We must find the probability of each context. Let us assume that there is an equal likelihood that the default value will indicate any one of the w wrenches. Then, the probability that it will indicate a wrench that the robot already has in its tool belt is simply

h/w.[6] Being in the second context (the wrench is in an open box) requires that we are not holding the correct wrench (probability $(w-h)/w$), *and* that the box containing the wrench is not closed (probability $(b-c)/b$). Similarly, the third and fourth contexts occur with probabilities $(w-h)/wb$ and $(w-h)(c-1)/wb$, respectively.

Using these probabilities together with the context-dependent recovery costs from Table 1, we define a function $I_D(b,w,h,c)$ to denote the expected incremental cost of using a default to recover from bad data given the values of b, w, h and c. We refer to this as an *incremental* cost since it computes the cost associated with a single recovery, and we will sum these costs later.

$$\begin{aligned} I_D(b,w,h,c) &= 1\cdot\frac{h}{w}+4\cdot\frac{(w-h)(b-c)}{wb}+6\cdot\frac{w-h}{wb}+10\cdot\frac{(w-h)(c-1)}{wb} \\ &= \frac{1}{bw}\left(bh+2(w-h)(2b+3c-2)\right). \end{aligned}$$

The same probabilities can be used for the incremental cost of attempting to recover via sensor use:

$$\begin{aligned} I_S(b,w,h,c) &= 4\cdot\frac{h}{w}+7\cdot\frac{(w-h)(b-c)}{wb}+7\cdot\frac{w-h}{wb}+13\cdot\frac{(w-h)(c-1)}{wb} \\ &= \frac{1}{bw}\left(4bh+(w-h)(7b+6c-6)\right). \end{aligned}$$

Tables 3 and 4 give expected recovery costs for values of h and c from 1 to 10 assuming $b=w=10$.

Recovery costs associated with bad data change as the agent proceeds through the various planning and execution phases. For instance, as the agent achieves more of its goals, it acquires more wrenches, reducing the likelihood of expensive recoveries where the robot does not already have the wrench. However, there are competing costs at work here. As the agent achieves more goals, more boxes are bolted shut, so the probability of having to reopen a box for the wrenches the agent has not collected increases.

As a comparison of these two recovery policies let us find their difference.

$$I_S(b,w,h,c)-I_D(b,w,h,c)=\frac{1}{bw}(3bw-2w+2h).$$

Note first that this difference does not depend upon c at all. Second, as b gets large, the difference approaches 3 for all values of w and h. This makes sense since from Tables 1 and 2 we can see that, in three of four contexts, recovering via sensor takes three more actions ((1) obtain a bolt, (2) sense it and (3) return it to its spot) than recovering via default. The only exception is where the needed wrench is in the box on which the robot is currently working in which case there is a helpful interaction between sensing and getting the wrench. As the number of boxes grows, the likelihood of this exception context shrinks.

Recovery via sensor is always more expensive than recovery via default, but we must remember that in the latter case we have already sensed the bolt size (with incorrect results) so a sensing cost has already been paid and evaluated as part of $C'(b,u,h)$. Furthermore, deciding to sense first and then recover if necessary by using a default can lead to excessive premature actions. Thus, it is incorrect to simply compare corresponding entries in each table to choose the least costly policy. The correct choice is also a function of sensor/default reliabilities, and the cost of recovering from premature actions for each.

There is of course a third type of recovery where human intervention is used. We treat human intervention as an expensive sensor operation that supplies information to the agent, but leaves the world unchanged. After the agent receives this information it can plan recovery actions. Again, context determines the exact operations and hence the cost. The actions taken are identical in all contexts to those used when recovering via default. Thus, we define

$$I_H(b,w,h,c)=I_D(b,w,h,c)+\eta$$

[6] One might think this ratio should actually be $(h-1)/(w-1)$, since we know one of the wrenches we are holding has already failed, but it is possible for recovery to yield the same incorrect value as the original attempt. Since BUMP does not detect such pointless recovery attempts, we do not use this knowledge in this calculation.

Wrenches Held	Boxes Closed									
	1	2	3	4	5	6	7	8	9	10
1	3.9	4.4	5.0	5.5	6.0	6.6	7.1	7.7	8.2	8.7
2	3.6	4.0	4.5	5.0	5.5	6.0	6.4	6.9	7.4	7.9
3	3.2	3.7	4.1	4.5	4.9	5.3	5.8	6.2	6.6	7.0
4	2.9	3.3	3.6	4.0	4.4	4.7	5.1	5.4	5.8	6.2
5	2.6	2.9	3.2	3.5	3.8	4.1	4.4	4.7	5.0	5.3
6	2.3	2.5	2.8	3.0	3.2	3.5	3.7	4.0	4.2	4.4
7	2.0	2.1	2.3	2.5	2.7	2.9	3.0	3.2	3.4	3.6
8	1.6	1.8	1.9	2.0	2.1	2.2	2.4	2.5	2.6	2.7
9	1.3	1.4	1.4	1.5	1.6	1.6	1.7	1.7	1.8	1.9
10	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 3: Expected Cost I_D (with $b = w = 10$).

Wrenches Held	Boxes Closed									
	1	2	3	4	5	6	7	8	9	10
1	6.7	7.2	7.8	8.3	8.9	9.4	9.9	10.5	11.0	11.6
2	6.4	6.9	7.4	7.8	8.3	8.8	9.3	9.8	10.2	10.7
3	6.1	6.5	6.9	7.4	7.8	8.2	8.6	9.0	9.5	9.9
4	5.8	6.2	6.5	6.9	7.2	7.6	8.0	8.3	8.7	9.0
5	5.5	5.8	6.1	6.4	6.7	7.0	7.3	7.6	7.9	8.2
6	5.2	5.4	5.7	5.9	6.2	6.4	6.6	6.9	7.1	7.4
7	4.9	5.1	5.3	5.4	5.6	5.8	6.0	6.2	6.3	6.5
8	4.6	4.7	4.8	5.0	5.1	5.2	5.3	5.4	5.6	5.7
9	4.3	4.4	4.4	4.5	4.5	4.6	4.7	4.7	4.8	4.8
10	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0

Table 4: Expected Cost I_S (with $b = w = 10$).

where, as before, η is the cost of a single human intervention.

Now we return to the question of finding $\delta_{b,w}$. In a few simple cases, $\delta_{b,w}$ is simply equal to an appropriate incremental cost. For example,

$$\delta_{3,5}(SD, D, I) = I_D(3, 5, 1, 1).$$

This is true since there is exactly one recovery, and at that point we know

1. the robot has exactly one wrench which the sensor told it to pick up (assuming it had zero to begin with), and
2. exactly one box is closed—the one the robot unsuccessfully tried to bolt—assuming every box was open initially.

However, in most cases we cannot determine before execution time the correct values for the arguments of the incremental cost functions. The value of c is not a problem, but we cannot determine h. We do not know before execution the values of the sensor readings that will be made and hence which wrenches will be collected. For example, after the event sequence SDI-DSI-DI-SI the robot might still have only two wrenches (all S's and D's indicate the same wrench, and the I's indicate one other wrench). Alternatively, the robot could have 10 wrenches if each attempt at closing a box used a new wrench. Thus, we cannot precisely calculate the context, nor the value for $\delta_{b,w}$.

3.5 Finding a Near-Optimal Policy

In this section we will use an approximation of expected cost for recovery from bad data, and this will allow us to obtain an $O(b \log b)$ algorithm for selecting the probabilistically optimal cost-based sensing policy. The approximation made here is that we ignore the context when calculating the cost of a recovery due to bad data and instead use a cost averaged over all contexts. An alternative method in which context is preserved and dynamic programming is exploited to control the inherent complexity is discussed thoroughly in [Krebsbach, 1993].

We first derive three new functions $I'_S(b,w)$, $I'_D(b,w)$ and $I'_H(b,w)$ denoting these average costs. The empirical study in [Krebsbach *et al.*, 1991] and the analysis in this paper made the domain restriction that a different wrench was needed for each box. This implies that in any reachable context $h \geq c$. We will use this fact in deriving $I'_D(b,w)$ and $I'_S(b,w)$ and $I'_H(b,w)$. The formula for $I'_D(b,w)$ is

$$\begin{aligned} I'_D(b,w) &= \frac{2}{w(w+1)} \sum_{h=1}^{w} \sum_{c=1}^{h} \frac{bh + 2(w-h)(2b+3c-2)}{bw} \\ &= \frac{2}{bw^2(w+1)} \sum_{h=1}^{w} [bh^2 + h(w-h)(3h+4b-1)] \\ &= \frac{3w^2 - 5w + 12bw - 6b + 2}{6bw}. \end{aligned}$$

The formula for $I'_S(b,w)$ is

$$\begin{aligned} I'_S(b,w) &= \frac{2}{w(w+1)} \sum_{h=1}^{w} \sum_{c=1}^{h} \frac{4bh + (w-h)(7b+6c-6)}{bw} \\ &= \frac{2}{bw^2(w+1)} \sum_{h=1}^{w} [4bh^2 + h(w-h)(3h+7b-3)] \\ &= \frac{w^2 - 3w + 10bw - 2b + 2}{2bw}. \end{aligned}$$

Also, $I'_H(b,w) = I'_D(b,w) + \eta$.

a	$\Delta_{b,w}(a,r,s)$
SDI	$(1-s)(I'_D(b,w)+(1-r)I'_H(b,w))$
SI	$(1-s)I'_H(b,w)$
DSI	$(1-r)(I'_S+(1-s)I'_H(b,w))$
DI	$(1-r)I'_H(b,w)$
I	0

Table 5: $\Delta_{b,w}(a,r,s)$.

This simplification enables us to approximate $\delta_{b,w}$ as a sum of the incremental recovery costs. First for any single decision point event sequence a_i, we define $\text{rec}(a_i)$ to be the set of recoveries of a_i. This is simply the set of non-initial resources in the sequence. For example, $\text{rec}(SDI) = \{D, I\}$. Now,

$$\delta_{b,w}(a_1,\ldots,a_b) \approx \sum_{i=1}^{b} \sum_{j\in\text{rec}(a_i)} I'_j(b,w).$$

This approximation of $\delta_{b,w}$ allows us to derive a very simple approximation for $K_{b,w}$. For example,

$$\begin{aligned} K_{2,w}(SDI,SDI) \;\approx\;& C'(2,2,0) \\ & + (1-s_1)(I'_D(2,w)+(1-r_1)I'_H(2,w)) \\ & + (1-s_2)(I'_D(2,w)+(1-r_2)I'_H(2,w)), \end{aligned}$$

and

$$\begin{aligned} K_{3,w}(DSI,SI,I) \;\approx\;& C'(3,1,1) \\ & + (1-r_1)(I'_S(3,w)+(1-s_1)I'_H(3,w)) \\ & + (1-s_2)I'_H(3,w). \end{aligned}$$

In each case we begin with C' and than add one term for each decision point reflecting the costs of recovering from bad data. This term is constructed as follows. If the first letter in the partial policy is D or S, find the probability that this attempt will fail. If the next letter is I, multiply this failure probability by I'_H. Otherwise, find the probability that the second attempt will fail, multiply this probability by I'_H, add to this the appropriate I' value for the second attempt and multiply by the probability of failure for the first attempt. Thus, DI yields $(1-r_i)I'_H$, and DSI yields $(1-r_i)(I'_S+(1-s_i)I'_H)$. The length of the entire approximating formula for K is $O(b)$ for b decision points.

In determining the policy for a particular decision point there are two crucial questions. First, should we start off with a sensor reading for this decision point? If yes, this probably implies additional premature actions. The exact number of extra premature actions caused by an S-initial policy at this decision point depends upon the total number of S-initial policies that are selected. Hence this part of the policy selection cannot be made locally.

Given a decision regarding the initial attempt, a second question is what options to use when recovering from bad data. If a decision has been made to start with a sensor reading, should the policy be SDI or SI? Similarly, if we decide not to start with a sensor should we choose DSI, DI or I?[7] Fortunately, this decision can be made locally. To see this, let us first introduce some convenient notation. Let $\Delta_{b,w}(a,r,s)$ be the expected cost of recovering from bad data during execution of a policy a for one decision point, assuming reliabilities r and s. The formulas for calculating $\Delta_{b,w}(a,r,s)$ based upon incremental recovery costs are shown in Table 5.

We can now write the expected cost of a policy as

$$K_{b,w}(a_1,\ldots,a_b) \;\approx\; C'(b,u,h) + \sum_{i=1}^{b} \Delta_{b,w}(a_i,r_i,s_i). \tag{6}$$

[7]Like a default, human intervention does not lead to premature actions. Thus, the I policy is grouped with DSI and DI.

Let us assume that some decision point a_j in formula (6) uses policy SDI. Now consider a modification of this policy where $a'_j = SI$:

$$K_{b,w}(a_1,\ldots,a'_j,\ldots,a_b) \approx C'(b,u,h) + \left(\sum_{i=1}^{b}\Delta_{b,w}(a_i,r_i,s_i)\right) - \Delta_{b,w}(a_j,r_j,s_j) + \Delta_{b,w}(a'_j,r_j,s_j). \tag{7}$$

Subtracting (7) from (6) gives the *improvement* (i.e., the decrease in cost):

$$\begin{aligned}\Delta_{b,w}(a_j,r_j,s_j) - \Delta_{b,w}(a'_j,r_j,s_j) &= (1-s_j)(I'_D(b,w) + (1-r_j)I'_H(b,w)) - (1-s_j)I'_H(b,w)\\ &= (1-s_j)(I'_D(b,w) - r_jI'_H(b,w)).\end{aligned}$$

We should prefer a'_j to a_j if this improvement is greater than or equal to zero. That is, the modified policy (with $a'_j = SI$) is at least as good as the original if

$$(1-s_j)(I'_D(b,w) - r_jI'_H(b,w)) \geq 0. \tag{8}$$

Since $s_j \leq 1$, this is true whenever $s_j = 1$ or $I'_D(b,w) \geq r_jI'_H(b,w)$. This can be determined independently of the policy selections for other decision points.

Similar reasoning can be used to select locally between the non-S-initial policies DSI, DI and I. We simply find the minimum of the three values $\Delta_{b,w}(DSI,r,s)$, $\Delta_{b,w}(DI,r,s)$ and η. If the first one is the minimum, select DSI; if the second, select DI; otherwise, select I.

We can express the above more succinctly by defining a function $score_{b,w}(a,r,s)$ where

$$score_{b,w}(a,r,s) = \begin{cases}\eta & \text{if } a = \text{I},\\ \Delta_{b,w}(a,r,s) & \text{otherwise.}\end{cases}$$

Then, for each decision point we consider the S-initial policy with the smaller score and the non-S-initial policy with the smallest score.

To obtain our policy selection algorithm, we start by considering a policy that consists only of the best S-initial components as determined above. Call this complete policy P. Now, consider m decision points within this policy. Let $p'_1,\ldots,p'_m$ be the best non-S-initial policies and $p_1,\ldots,p_m$ be the best S-initial policies for these m decision points. Furthermore, let P' be the complete policy that results from substituting $p'_1,\ldots,p'_m$ for $p_1,\ldots,p_m$ in P. The improvement in using P' instead of P is

$$K_{b,w}(P) - K_{b,w}(P') \approx \left(\sum_{i=1}^{m}[score_{b,w}(p_i,r_i,s_i) - score_{b,w}(p'_i,r_i,s_i)]\right) + C'(b,b,0) - C'(b,b-m,h^*), \tag{9}$$

where h^* is the number of I policies in $p'_1,\ldots,p'_m$. Note that

$$C'(b,b,0) - C'(b,b-m,h^*) = m\sigma - h^*\eta + \pi[\rho(b,b) - \rho(b,b-m)].$$

If we wish to use m non-S-initial policy components, it should be the m components that maximize the value of the improvement shown in (9). Given that b and m are fixed for the moment, this amounts to maximizing

$$\left(\sum_{i=1}^{m}[\Delta_{b,w}(p_i,r_i,s_i) - \Delta_{b,w}(p'_i,r_i,s_i)]\right) - h^*\eta \tag{10}$$

To find the best m components we do the following. For each decision point, i, find the difference between the scores of the best S-initial component, p_i, and best non-S-initial component, p'_i, (S-initial minus non-S-initial). This in effect calculates one term of the sum shown in (10). Call this difference the *cost-competitiveness* of the non-S-initial component. The full set of cost-competitiveness values can be found in time $O(b)$ for b decision points. Next, sort the decision points in decreasing order of cost-competitiveness ($O(b\log b)$). The m components maximizing (10) will be the first m components in the resulting list. We finish the policy selection process by deciding *how many* non-S-initial components to use. The $O(b)$ algorithm in Figure 3 is applied to the sorted list of decision points to determine this number. Thus, the complete algorithm takes time $O(b\log b)$.

```
BestM := 0;
BestImpr := 0;
Sum := 0;
for m := 1 to b do
    begin
        if $p'_m$ = 'I' then $h^* := h^* + 1$;
        Sum := Sum + $\Delta_{b,w}(p_m, r_m, s_m) - \Delta_{b,w}(p'_m, r_m, s_m)$;
        Impr := Sum + $C'(b, b, 0) - C'(b, b - m, h^*)$;
        if Impr > BestImpr then
            begin
                BestImpr := Impr; BestM := M
            end
    end
```

Figure 3: Algorithm for best number of defaults.

Method	*All SDI*	*Defaults Algorithm*	*Dynamic Programming*	*Exhaustive*			*Domain Dependent*
				Best	Worst	Average	
Cost	67.0	57.2	57.4	57.4	80.7	70.0	65.0

Table 6: Comparison of expected costs with $b = 5, r_i = 0.4, s_i = \{.5, .6, .7, .8, .9\}$ and $\eta = 10$.

3.6 Comparison of Performance

Tables 6 and 7 contain expected costs for two problems.

1. Five boxes with all default reliabilities equal to 0.4, and the five sensor reliabilities equal to 0.5, 0.6, 0.7, 0.8 and 0.9, respectively.

2. Ten boxes with all default reliabilities equal to 0.2 and all sensor reliabilities equal to 0.7.

3.6.1 Domain-Independent Strategies

Five different methods were used to derive the cost figures displayed. The first of these uses a policy that consists entirely of SDI. The second uses the policy selected by the algorithm in Figure 3.

The third uses the dynamic programming approach outlined in [Krebsbach, 1993], while the fourth provides best, worst, and average costs derived from checking all possible solutions. There are a few things to make mention of in the third and fourth cases. First, the dynamic programming method is a polynomial time version of the exhaustive method, both of which guarantee an optimal solution. For this reason, these two methods come up with the same sensor strategy and thus

Method	*All SDI*	*Defaults Algorithm*	*Dynamic Programming*	*Exhaustive*			*Domain Dependent*
				Best	Worst	Average	
Cost	147.4	134.6	133.4	*	*	*	130.0

Table 7: Comparison of expected costs with $b = 10, r_i = 0.2, s_i = 0.7$ and $\eta = 10$. A * indicates a value that we were unable to compute.

produce the same (minimal) cost. The exhaustive method is included both to provide and upper bound and an average, (which is not possible with dynamic programming, since many paths are pruned) and finally to emphasize the point that it was not possible for us to obtain an answer for even a ten-box problem exhaustively. Asterisks are used to indicate this in Table 7.

Some explanation is necessary for the fact that the cost in the Defaults Algorithm column of Table 6 is smaller than the Dynamic Programming cost and the Best-case Exhaustive column which describes the optimal expected cost over all strategies. The cost in the Defaults Algorithm column is calculated using the same approximations for the cost of recovery from bad data as are used in the algorithm itself while the Exhaustive search costs use the actual expected costs of recovery from bad data with the full recovery context taken into account. It is important to note that the same sensing policy was obtained in both cases, so the Defaults Algorithm did find the optimal strategy. Furthermore, the closeness of the costs in these two columns suggests that the approximations used in the Defaults Algorithm result in a cost that is a reasonable approximation to the actual expected cost.

3.6.2 Domain-Dependent Strategies

The final column of these tables gives an estimate assuming a special-purpose domain-dependent technique such as Sense Before Closing. There are different ways to design such a strategy. The data in the table assumes the following simple two-pass method. One pass is made to sense all bolts in all boxes and pick up every wrench found (under the assumption that they will all be needed). A second pass is then made to perform all actions to achieve the goals. While this method eliminates premature action completely, costs due to bad data are still present (we assume 10 actions to recover in 30% of the cases, based on the sensor reliabilities given here). In addition, SBC produces many more goto actions and, more importantly, is extremely dependent on the number of wrenches included in the domain, which is conservatively assumed here to be the same as the number needed. Even with these generous assumptions, it can be seen that this domain-dependent strategy only performs on par with the best domain-independent strategies.

In reviewing this line of research, a number of our colleagues have asked the following question, in some form or another:

> "Why explore general purpose strategies like Stop and Execute (SE) and Continue Elsewhere (CE) in a domain like the Tool Box World when they are known to lead to premature action in some cases? Why not use a strategy like Sense Before Closing (SBC)? Since by definition such a strategy would obtain *all* sensor data before closing *any* boxes, there is no chance for premature action. Why not eliminate premature action altogether?"

The answer to this question depends upon what you think is an interesting research question to ask. The rest of this section is intended to demonstrate how easily a specific strategy like SBC breaks down with relatively slight perturbations to the problem. It should be noted however that the authors of this paper spent many hours exploring exactly how generally applicable strategies like SBC really are. The following scenarios are exemplary of our findings.

Scenario One: Adding Wrenches. Suppose we now take the ten-box problem mentioned previously, except that the number of wrenches in the domain is 1000 instead of b. Since even the SBC (two pass) strategy does not provide the agent with information about *which* wrenches to pick up in time to be useful, the agent would have to pick up 1000 wrenches to guarantee no premature action, but at what a cost! This would imply at least 990 unnecessary wrench pickup actions alone. For matter's of comparison, Krebsbach's cost formulas [Krebsbach, 1993] would predict that adding 990 wrenches to the problem implies not 990 new actions, but only an additional 22.0 actions to the optimal domain-independent strategies (155.4 vs. 133.4 actions).

Even in the worst case with the SE strategy (assuming every sensor and default fails, and that human interaction is very expensive) we will be in much better shape than SBC in this case. In this reasonable problem, SBC is equivalent to a mechanic starting his day by driving to auto parts stores and filling a truck with all of the parts and tools that could potentially be needed on that day without a clue as to the jobs to be performed.

It should be noted that a *three pass* strategy would be more appropriate to combat the problems associated with the proliferation of wrenches. With this strategy, a middle pass could be used to collect only the 10 matching wrenches. Here the penalty is in extra goto actions. The number of goto operations would be between 19 and 30 inclusive, depending on how the wrenches are distributed and how sophisticated the algorithm is at minimizing gotos.[8] The number of gotos alone is likely to be enough to overwhelm what Krebsbach [Krebsbach, 1993] predicts as premature action cost for this problem, which of course is all that is being saved.

Scenario Two: Expensive Gotos. The brittleness of SBC is further demonstrated if we allow the cost of gotos to vary with an aspect of the domain (e.g., distance between boxes). Suppose we again are faced with the problem described in scenario one, except that costs associated with travelling between boxes (i.e., gotos) depend on distances between boxes (a more reasonable assumption) and that such costs are generally fairly high in relation to the unit cost of other actions. Since SBC depends heavily on visiting each box two or three times, SBC could be badly inefficient. SBC is specific to a very small class of problems, and becomes largely inefficient with small perturbations to the problem; a bad quality in a strategy.

Scenario Summary The point we wish to make with this example is not simply that one special-purpose strategy works poorly with one particular domain assumption (e.g., number of wrenches) or cost assumption (e.g., expensive gotos). The principle is that the more general techniques presented here allow the user to take the specific constraints of one of a large class of problems into account by building them into the cost formulas. For example, SBC is good for a small class of problems in which recovering from premature action is very expensive, (since it eliminates premature action completely). However, by altering the cost formulas to reflect this (either analytically or empirically), our approach would work as well, and for a much larger class of problems. By identifying and accounting for the factors which influence sensor decisions in the cost formulas, specific strategies for every problem and variation thereof can be largely avoided, allowing a technique like dynamic programming to minimize these costs automatically.

3.6.3 Application to Other Domains

Our principle goal in this paper is not to provide the precise formulas for the Tool Box World, but rather to gain a sufficient understanding of the factors that influence policy selection. Although we have analyzed these factors mathematically, the algorithms that have come out of this understanding can be used in other domains without performing a similar mathematical analysis. To apply the algorithm in another domain, the following things must be true:

1. There is a way to determine the decision points prior to planning.
2. $\rho(b, u)$ is known or can be approximated.
3. $I'_S(b, w), I'_D(b, w)$ and $I'_H(b, w)$ are known or can be approximated.
4. The constants β, σ, η and π are known or can be approximated.

Approximations for items (2)–(4) can all be obtained empirically by actually running the planner on sample problems. This would allow the use of the algorithm in most realistic domains that are too difficult or time-consuming to analyze mathematically.

[8]The middle pass could probably be optimized somewhat by trying to pick up a wrench on the first pass if we already know that we need it. The degree to which this can be done depends upon the placement of the wrenches.

4 A Success-Based Analysis

In this section we analytically determine the expected success rates for the Tool Box World. We then use this analysis to obtain an algorithm for optimizing success rates by selecting a sensing policy. This method is appropriate when the robot is unable to recover from premature actions and bad data. The result of this analysis will be the demonstration that the domain-independent extensions studied in this paper are not sufficient when success is the criterion and domain-dependent strategies are most likely required.

4.1 Predicting Success Rates

In this analysis we make the same assumptions as made in the cost analysis. Namely, we assume the top level goals are reordered, as described in Section 2 so as to maximize the average success rate, and we use the Stop and Execute control strategy. As a starting point we consider cases where there are b boxes and exactly 2 unknowns. We then generalize this result to the success rate for b boxes and u unknowns which will be denoted by $Q(b, u)$.

Let us reconsider our examples from Section 3.2.1. To review, there are 3 tool boxes S, T and U that are to be bolted closed with bolts b_s, b_t and b_u, respectively. The sizes of these three bolts are 4, 5 and 6, respectively with only b_u known. The goals are considered by the planner in the order S, T then U. Also, b_s is in S, b_t is in T and b_u is in U.

In our earlier empirical study of these problems we found that out of the 27 cases in this example there are 10 failures, and all of them have the following characteristic: either wrench 5 or wrench 6 (or both) is prematurely locked in box S. In both cases the premature action must occur before bolt b_t is sensed since once this sensor operation occurs the planner has all necessary information and a complete plan is found. BUMP can succeed if something causes it to sense b_t before closing S. This will happen whenever the robot must travel to box T (the location of b_t) before closing S, and this in turn will happen if wrench 4 is in T.

We can concisely describe the failures in this case by the logical formula

$$4 \notin T \wedge (5 \in S \vee 6 \in S)$$

Negating this gives the successful cases

$$4 \in T \vee (5 \notin S \wedge 6 \notin S)$$

The first disjunct, $4 \in T$ is true in 1/3 (= 9/27) of the problems. When it is false, BUMP still succeeds if the second disjunct is true. This combination occurs in $(2/3)^3 = 8/27$ of the cases, all of which are distinct from the previous 1/3. Given 27 problems in the three box domain, we have accounted for the 17 successful cases.

Now let us extend our analysis to the case where there are b boxes and 2 unknowns. Using our earlier notation, BUMP will be successful if and only if none of the wrenches are bolted inside $B(g_1)$. There are two ways to guarantee this. The first is that $W(g_1) \in B(g_2)$ since this will result in the construction of a complete plan before any boxes are closed. The second is that none of the wrenches $W(g_2), W(g_3), \ldots, W(g_b)$ are in $B(g_1)$. The first of these conditions is true in b^{b-1} of the b-box problems. When this condition is false (i.e., $W(g_1) \notin B(g_2)$), there are $b-1$ possible places for $W(g_1)$. Since none of the other wrenches can be in box $B(g_1)$ there are also $b-1$ places for each of those wrenches. This gives success in $(b-1)^b$ problems (all distinct from the previous ones). Adding and then dividing by the number of b-box problems gives us the success rate:

$$Q(b, 2) = \frac{b^{(b-1)} + (b-1)^b}{b^b} = \frac{1}{b} + \left(\frac{b-1}{b}\right)^b. \tag{11}$$

Figure 4 contains a plot of $Q(b, 2)$.

It is interesting to examine what happens as b grows with u fixed at 2. Taking the limit of $Q(b, 2)$ as b approaches infinity yields the following:

$$\lim_{b \to \infty} Q(b, 2) \quad = \quad \lim_{b \to \infty} \left[\frac{1}{b} + \left(\frac{b-1}{b}\right)^b\right]$$

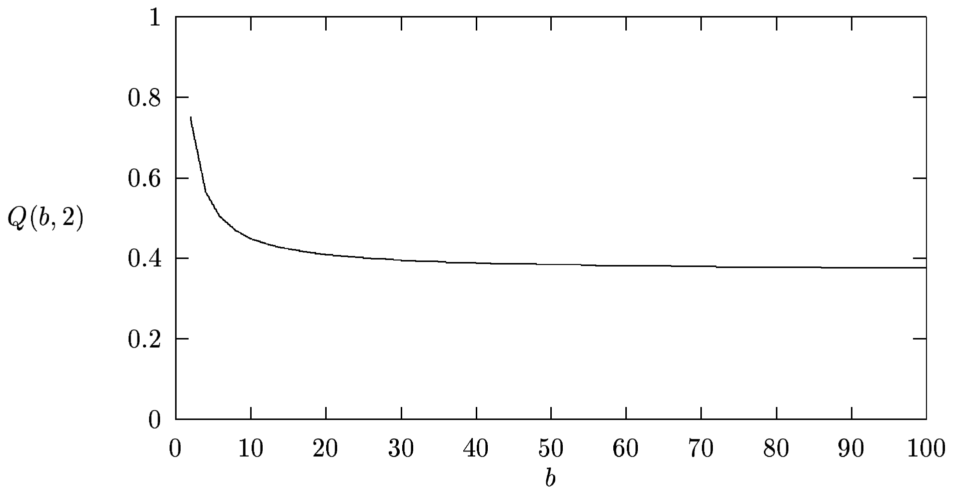

Figure 4: Graph of $Q(b,2)$.

$$= \lim_{b\to\infty}\left(\frac{b-1}{b}\right)^b.$$

Using the substitution $b = 1/x$ we convert this limit to

$$\lim_{x\to 0+}(1-x)^{1/x}. \tag{12}$$

Taking the natural logarithm of (12) and using L'Hôpital's rule we get

$$\begin{aligned}\lim_{x\to 0+}\ln(1-x)^{1/x} &= \lim_{x\to 0+}\frac{-1}{1-x}\\ &= -1.\end{aligned}$$

Thus,

$$\lim_{b\to\infty} Q(b,2) = e^{-1} \approx .3678.$$

As with cost, this limit suggests that in this domain it is not the number of boxes that makes a problem difficult, but the number of unknowns. Having more boxes interacts with this for small values of b, but past a certain point it makes virtually no difference how many boxes there are, and with only 2 unknowns we are guaranteed of a success rate of at least 36%.

4.2 Generalizing to $Q(b,u)$

In generalizing our result from the previous section it is convenient to talk about the number of successes in addition to the success rate. We will use the notation $S(b,u)$ to refer to the number of successes (considering the entire problem space) with b boxes and u unknowns. There is a simple relationship between S and Q.

$$S(b,u) = Q(b,u)\cdot b^b. \tag{13}$$

With zero or one unknown there are no failures due to premature actions since no boxes are closed prior to completion of the plan. So

$$\begin{aligned}S(b,0) &= b^b & (14)\\ S(b,1) &= b^b. & (15)\end{aligned}$$

	Unknowns								
Boxes	2	3	4	5	6	7	8	9	10
2	.750								
3	.630	.370							
4	.566	.281	.168						
5	.528	.241	.121	.072					
6	.502	.218	.100	.050	.030				
7	.483	.203	.088	.040	.020	.012			
8	.469	.192	.081	.035	.016	.008	.005		
9	.458	.185	.076	.032	.014	.006	.003	.002	
10	.449	.179	.072	.030	.013	.005	.003	.001	.001

Table 8: $Q(b,u)$ for $2 \leq b \leq 10$ and $2 \leq u \leq b$.

Using our result (11) from the previous section, along with formula (13), we also know

$$S(b,2) = b^{b-1} + (b-1)^b. \tag{16}$$

We now derive a recurrence relation for $S(b,u)$. Consider g_1, and assume $u > 2$. If none of the wrenches for goals $g_2, g_3, \ldots g_b$ are in $B(g_1)$, then no matter where $W(g_1)$ is located BUMP will not lock any needed wrenches into $B(g_1)$. Assuming that the wrenches for the remaining $b-1$ boxes ($u-1$ of them with unknown information) are arranged so as to avoid any premature actions, BUMP will succeed on the problem. Using recursion, this gives $b \cdot S(b-1,u-1)$ problems on which BUMP will succeed. The only other cases that do not involve locking a wrench in $B(g_1)$ are when $W(g_2) \in B(g_1) \land W(g_1) \in B(g_2)$ (i.e., a *flipped* condition). Obviously, none of the wrenches for the other goals can be in $B(g_1)$ since this would produce a 2-back failure. Similarly, none of them can be in $B(g_2)$ since $W(g_2) \in B(g_1)$ and

1. having $W(g_3)$ in $B(g_2)$ would produce a half-flipped failure, and
2. having any of the wrenches $W(g_4), \ldots, W(g_b)$ in $B(g_2)$ would produce a *2-back* failure.

Thus, with the first two wrenches flipped, BUMP will succeed whenever the wrenches for the remaining $b-2$ boxes (with $u-2$ having unknown information) are in a successful configuration involving only those $b-2$ boxes. This is another $S(b-2,u-2)$ cases. Combining these two distinct classes of success gives

$$S(b,u) = b \cdot S(b-1,u-1) + S(b-2,u-2) \tag{17}$$

where $u > 2$. The base cases for this recurrence are given in (15) and (16).

From this we can find a recurrence relation for the expected success rate for any b box, u unknown tool box problem.

$$\begin{aligned} Q(b,u) &= \frac{S(b,u)}{b^b} \\ &= \frac{b \cdot S(b-1,u-1) + S(b-2,u-2)}{b^b} \\ &= \frac{b(b-1)^{b-1}Q(b-1,u-1) + (b-2)^{b-2}Q(b-2,u-2)}{b^b}. \end{aligned}$$

The values of $Q(b,u)$ for $b = 2,\ldots,10$ and $u = 2,\ldots,b$ are shown in Table 8.

4.3 Selecting a Sensing Policy

Now we must account for the unreliability of sensors and default values. As before, let r_i be the reliability of a default value and s_i the reliability of a sensor reading. We assume that if incorrect default information is used, the robot will eventually detect an execution time error. This will necessitate some amount of execution time error recovery, and the resulting execution will certainly be inefficient. For irreversible actions, we consider this a failure, so the decrease in premature actions by having extra "known" information must be adjusted by the reliability of that information. A similar point can be made regarding sensor reliability.

In [Krebsbach *et al.*, 1991, Krebsbach *et al.*, 1992] we outlined an algorithm for selecting the sensing policy that optimizes the success rate given that default and sensor information might be erroneous. This algorithm consisted of evaluating the expected success rate for all combinations of default/sense decisions and taking the combination with the largest value. As an example, for the 3 box problem the expected success rates of each of the eight sensing policies are:

$$\{r_1 r_2 r_3 Q(3,0), r_1 r_2 s_3 Q(3,1), r_1 s_2 r_3 Q(3,1), s_1 r_2 r_3 Q(3,1),$$
$$r_1 s_2 s_3 Q(3,2), s_1 r_2 s_3 Q(3,2), s_1 s_2 r_3 Q(3,2), s_1 s_2 s_3 Q(3,3)\}$$

Unfortunately, the complexity of this algorithm is $O(b2^b)$ where b is the number of decision points. In this section we further analyze this problem and present an $O(b \log b)$ algorithm for selecting the sensing policy that maximizes the expected success rate. This algorithm is similar to the one presented earlier for cost-based policy selection, although the current one is somewhat easier to understand and does not require any approximations.

As a starting point, consider the case where we decide to use a sensor to determine the wrench required for each of b decision points. The expected success rate is

$$Q(b,b) \prod_{i=1}^{b} s_i. \tag{18}$$

Now consider the effect on the success rate of using a default for one of those b decision points, d. This implies one fewer unknown and a change in reliability. So the new success rate is

$$Q(b,b-1) \frac{r_d}{s_d} \prod_{i=1}^{b} s_i. \tag{19}$$

Dividing (19) by (18) gives the factor of improvement

$$\frac{r_d}{s_d} \cdot \frac{Q(b,b-1)}{Q(b,b)}.$$

There are two important contributions to this factor of improvement. One is the comparative reliabilities, r_d and s_d, of the default and sensor, respectively. We will call the ratio r_d/s_d the *competitiveness* of the default. The other factor represents the improvement gained from using one default (as compared to zero). This factor will be the same no matter which default we use. From this it is clear that if we use exactly one default, it should be the one with the maximum competitiveness ratio. Taking any other default instead could never lead to a larger factor of improvement.

More generally, the factor of improvement in taking m defaults $i_1, \ldots, i_m$ is

$$\frac{r_{i_1}}{s_{i_1}} \frac{r_{i_2}}{s_{i_2}} \cdots \frac{r_{i_m}}{s_{i_m}} \cdot \frac{Q(b,b-m)}{Q(b,b)}.$$

Again, there are two contributions to this improvement:

1. the product of the competitiveness ratios of the m defaults, and

2. the improvement $Q(b,b-m)/Q(b,b)$ from taking m defaults.

Since the second contribution is independent of the defaults that are selected, it is best to take the m defaults with the largest competitiveness ratios. Substituting other defaults could only reduce the improvement factor. Now the problem of deciding *which* defaults to take is reduced to the problem of deciding *how many* defaults to take. This is a much easier problem.

```
BestM := 0;
BestImpr := 1;
Prod := 1;
for m := 1 to b do
    begin
        Prod := Prod × $r_m/s_m$;
        Impr := Prod × $Q(b, b-m)/Q(b, b)$;
        if Impr > BestImpr then
            begin
                BestImpr := Impr; BestM := M
            end
    end
```

Figure 5: Algorithm for best number of defaults.

We first add the following convention: the decision points are sorted in order of decreasing competitiveness. That is,

$$\frac{r_1}{s_1} \geq \frac{r_2}{s_2} \geq \ldots \geq \frac{r_b}{s_b}.$$

Now, the optimal sequence of sense/default decisions is defined as follows. Take defaults $d_1, d_2, \ldots d_m$ for $0 \leq m \leq b$ where m produces the maximum value for

$$\left(\prod_{i=1}^{m} \frac{r_i}{s_i}\right) \frac{Q(b, b-m)}{Q(b, b)}.$$

Sorting the decision points takes time $O(b \log b)$. Figure 5 shows an $O(b)$ algorithm to find the number of defaults that produces the maximum improvement (assuming the Q-values have been precomputed). Combining the sort with the algorithm in Figure 5 yields an $O(b \log b)$ algorithm.

In summary, if we know the Q-values and the reliabilities of the sensors and defaults, we can find the sensing policy that optimizes the expected success rate. This can be done in time $O(b \log b)$. Furthermore, the primary factor that determines the usefulness of a default is its competitiveness ratio, which can be calculated prior to planning and appears to be a more useful measure than the reliabilities themselves.

4.4 Practical Implications

Although the algorithm described in Section 4.3 produces the optimal sequence of sense/default decisions, it does not guarantee a very high success rate. For example, consider the 6 box problem. For simplicity let us assume that $r_i = \rho$ and $s_i = \sigma$ for all i $(1 \leq i \leq 6)$. In this case, all of the competitiveness ratios are ρ/σ. The optimal policy is determined by the value of m that maximizes

$$\left(\frac{\rho}{\sigma}\right)^m \frac{Q(b, b-m)}{Q(b, b)}.$$

For $\rho = 0.4$ and $\sigma = 0.8$ the optimal policy has $m = 4$. The expected success rate will be $\rho^4\sigma^2 Q(6, 2) = 0.8\%$. We have given BUMP an extremely difficult task. It is missing six pieces of crucial information, has sensors that work only 80% of the time, and we asked it to solve the problem without making *any* mistakes. It cannot do this well using only the domain-independent techniques explored in this paper. Special-purpose domain-dependent strategies are almost certainly needed.

5 Related Research

Traditional approaches to planning assume that the planner has access to all of the world information needed to develop a complete, correct plan—a plan which can then be executed in its entirety. Unfortunately, information about the world is not always available at plan time.

Several solutions have been proposed to deal with incomplete knowledge. They range from eliminating planning altogether in favor of reactive planning [Brooks, 1986] or situated systems [Agre and Chapman, 1987, Kaelbling, 1988], to combining reactivity and planning [Georgeff and Lansky, 1987, Firby, 1987, Wilkins *et al.*, 1994], to verifying, before execution, the executability of plans and adding sensing whenever needed to reduce the uncertainty [Doyle *et al.*, 1986], to preplanning for every contingency [Schoppers, 1987, Peot and Smith, 1992].

Since our work is an extension of classical planning, we will limit our comparison to solutions that are fundamentally based on planning.

5.1 Planning with Incomplete Information

There have been a number of efforts to characterize the role of information in the context of classical planning. Moore [1985] first focused attention on the role of the knowledge of an agent in planning and acting to achieve a goal. When the knowledge is incomplete, the agent has to check whether it has the knowledge necessary to carry out the plan, and to reason about how to obtain the missing information.

The need to include actions to acquire information has prompted a number of extensions to methods for representation of actions. For instance, Pednault [Pednault, 1989] allows actions with context-dependent effects [Pednault, 1989]. A language has been proposed [F. Giunchiglia, 1994] to represent and reason about actions and plans that are not guaranteed to succeed.

Etzioni and coworkers [1992] make a distinction between *information goals*, i.e., goals to gather information, and *information gathering actions*, i.e., actions that change the world and, as a result, force the world to be in a known state. They show how both can be represented by annotating preconditions and postconditions of STRIPS operators, and they present a provably correct planning algorithm based on their representation. We have not addressed representation issues because for the domain we examined STRIPS-like operators are sufficient.

The XII planner [Golden *et al.*, 1994] allows planning with incomplete information. It takes a middle ground between the traditional *closed world assumption* (which states that information not known is false) and the *open world assumption* (which states that information not explicitly known is unknown). XII uses a *local closed world* database to store the information that has been acquired. No specific control strategy is used for deciding when to execute. It appears that, as soon as XII detects the need for missing information, it plans on acquiring it and goes ahead to execute the plan. The interleaving of execution with planning makes XII incomplete when actions are irreversible. Our work proposes an algorithm for evaluating and selecting rationally a sensing policy, given some criteria for plan quality. We have proposed and analyzed different control strategies to be used by the planner in deciding when to sense, and we have shown how the choice of the control strategy depends on the plan quality criteria.

Genesereth and Nourbakhsh [1993] describe rules for pruning the graph of possible states when the information is incomplete. Their analysis focuses on incomplete information about the initial state. For non-trivial problems the state graph becomes so large that, even with massive pruning, finding a solution might be infeasible in practice. In BUMP we use the more traditional approach to planning in which search is performed in the space of partial plans as opposed to the state-space graph, and so the search space never grows so large to be unmanageable.

Conditional planning [Peot and Smith, 1992] and contingency planning [Schoppers, 1987] create a plan that will work under all circumstances by creating tree structured plans that branch depending on sensor outcomes. Conditional planning is a way of avoiding the problem of interleaving planning with execution. Unfortunately, the search space becomes prohibitively large, and this makes this approach impractical for any reasonably complex problem.

5.2 Interleaving Planning with Execution

Interleaving planning with execution has been proposed by many as a way of getting around the issue of missing or uncertain information [McDermott, 1978].

Our work has been inspired, among others, by the work of Turney and Segre [1989], who studied strategies for alternating between improvising and planning. Since sensing is assumed to be expensive, their system prefers actions with the fewest sensor requests first. Their results show that the quality of the heuristic improvisation strategy has the largest effect on the quality of the solution.

Wilkins [1994] interleaves planning with execution, but uses two different systems, one for planning (SIPE-2) and one for reactive execution (PRS-CL). His interest is in designing a complex system that can react to changes in the environment and replan. He does not address explicitly the issue of when and how to acquire missing information, and what is the effect of this choice on the quality of the plan produced.

5.3 Planning with Uncertainty

There is a difference between planning with uncertainty and planning with missing information like we do. Most methods of planning with uncertainty resort to monitoring information using sensors to keep the uncertainty bounded [Doyle *et al.*, 1986], and often use stochastic models [Hager and Mintz, 1991, Cassandra *et al.*, 1994, Draper *et al.*, 1994]. For instance, Hager and Mintz [1991] have proposed methods for sensor planning based on probabilistic models of uncertainty. Draper et al [Draper *et al.*, 1994] have introduced a variety of actions, such as information producing actions, imperfect sensing, and informational dependencies in the context of classical planning, and extended the applicability of planning by creating probabilistic planning algorithms.

5.4 Decision Theoretic Methods

The idea of planning to gather information is common in decision analysis. A number of authors have proposed decision theoretic approaches to decide what and when to sense to produce optimal (or near optimal) plans. Horvitz [1989] proposes a general model for reasoning under scarce resources that is based on decision theory. Chrisman and Simmons [1991] produce near optimal cost plans by using Markov Decision Processes to decide what to sense. Hansen [Hansen, 1994] incorporates sensing costs in the framework of stochastic dynamic programming.

Abramson [1991] casts sensory integration as a decision problem and presents a formula for deciding how often to sense depending on the rate of change of the environment. He assumes that the plan is executed in an environment in which errors occurs "spontaneously" and so the problem is that of finding what fixed rate of sensing is optimal. The problem we address in our research is how and when to sense to acquire missing information, not how and when to monitor.

Wellman and Doyle [Wellman and Doyle, 1992] propose modular utility functions for decision-theoretic planning. Modular functions allow specifying preferences, and composition methods allow combining them. Unfortunately, constructing utility functions from goals requires significant modeling of the problem domain.

Haddaway and Hanks [Haddawy and Hanks, 1993] propose a method for combining decision theory with planning. They assume a context in which goals might be only partially satisfied and have both a temporal and atemporal component. They define utility of goals and provide a rich utility model, but they have not yet incorporated the utility models into a planning algorithm.

Work on extending the approach presented in this paper to use dynamic programming is described in [Krebsbach, 1993]. A tree of possible sensing policies is generated *offline* using dynamic programming, the optimal sensor decisions are cached at each level, and subsequently actual world states are used as indexes into this structure to make a rational sensor choice *online*. The method is polynomial in both space and time.

6 Concluding Remarks

Planning is a common and important human activity. Its essence is to predict the future and base future actions on those predictions and the goals to be achieved. The advantage of planning over simply reacting then, is to discover potential conflicts between the goals and the environment *in time* to modify the plan of action (or perhaps the goals). We believe then, that planning is not only a convenient, but an essential ingredient in the solution of a large class of problems.

This paper is intended to serve two purposes. The first is to demonstrate that important advantages of planning need not disappear the moment we relax the classical assumption of an omniscient (all-knowing) planner. We have shown this in a general way by endowing a fairly typical classical planner with the ability to interleave planning and execution. This modified planner is able to solve problems involving environmental uncertainty which classical planners, by definition, cannot.

Moreover, we identify what we consider to be the two major costs associated with the approach, *premature action* and *bad data*, and provide a thorough analysis of these costs with regard to our demonstration domain, the Tool Box World. While the analytic results we have obtained depend on characteristics of this particular domain, we believe that the difficulties associated with premature action and bad data are central to a large class of planning problems, especially those exhibiting the characteristics outlined in Section 2.5.

The second purpose of the paper is to demonstrate how decisions involving adding sensing operations to a plan can be made *rationally* and in a *computationally feasible* way. We present a number of algorithms toward this end. Through these algorithms we propose that making rational sensing decisions involves defining a criterion or criteria one wishes to optimize (e.g., execution cost), and then identifying the trade-offs inherent in the domain between the decisions made and their effect on those criteria. Some of the effects may be treated in a context-free way (e.g., premature action) while others may be more context-dependent (e.g. bad data). Again, while the cost analyses here are specific to our domain, we intend the algorithms to be more general so as to provide practical suggestions for computing costs feasibly in other domains, as well as encouraging further research in the area of interleaved planning.

Acknowledgement

We wish to thank Mark Boddy for useful and influential discussions about topics in this paper, and Illah Nourbakhsh for comments on earlier versions of this work. We would also like to thank John Schue for his assistance in computing the limits of functions.

References

[Abramson, 1991] Bruce Abramson. An analysis of error recovery and sensory integration for dynamic planners. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 744–749, 1991.

[Agre and Chapman, 1987] Philip Agre and David Chapman. Pengi: A theory of activity. In *Proc. of AAAI-87*, pages 268–272, Washington, July 1987.

[Brooks, 1986] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, March 1986.

[Cassandra *et al.*, 1994] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *Proc. Nat'l Conf. on Artificial Intelligence*, volume 2, pages 1023–1028, 1994.

[Chrisman and Simmons, 1991] Lonnie Chrisman and Reid Simmons. Sensible planning: Focusing perceptual attention. In *Proc. of AAAI-91*, Los Angeles, CA, 1991.

[Doyle *et al.*, 1986] Richard Doyle, David Atkinson, and Rajkumar Doshi. Generating perception requests and expectations to verify the execution of plans. In *Proc. of AAAI-86*, pages 202–206, Philadelphia, 1986.

[Draper *et al.*, 1994] Denise Draper, Steve Hanks, and Daniel Weld. Probabilistic planning with information gathering and contingent execution. In *International Conference on AI Planning Systems*, 1994.

[Etzioni *et al.*, 1992] Oren Etzioni, Steve Hanks, Daniel Weld, Denise Draper, Neal Lesh, and Mike Williamson. An approach to planning with incomplete information. In *Third International Conference on Knowledge Representation and Reasoning*, 1992.

[F. Giunchiglia, 1994] P. Traverso F. Giunchiglia, L. Spalazzi. Planning with failure. In *International Conference on AI Planning Systems*, 1994.

[Firby, 1987] R. James Firby. An investigation into reactive planning in complex domains. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 202–206, Seattle, 1987.

[Genesereth and Nourbakhsh, 1993] Michael R. Genesereth and Illah R. Nourbakhsh. Time-saving tips for problem solving with incomplete information. In *Proc. of AAAI-93*, 1993.

[Georgeff and Lansky, 1987] Michael P. Georgeff and Amy L. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA, 1987.

[Golden *et al.*, 1994] Keith Golden, Oren Etzioni, and Daniel Weld. Omnipotence without omniscience: efficient sensor management for planning. In *Proc. Nat'l Conf. on Artificial Intelligence*, volume 2, pages 1048–1054, 1994.

[Haddawy and Hanks, 1993] Peter Haddawy and Steve Hanks. Utility models for goal-directed decision-theoretic planners. Technical Report 93-06-04, Dept. of Computer Science and Engineering, University of Washington, 1993.

[Hager and Mintz, 1991] G. Hager and M. Mintz. Computational methods for task-directed sensor data fusion and sensor planning. *International Journal of Robotics Research*, 10:285–313, 1991.

[Hansen, 1994] Eric A. Hansen. Cost-effective sensing during plan execution. In *Proc. Nat'l Conf. on Artificial Intelligence*, volume 2, pages 1029–1035, 1994.

[Hendler *et al.*, 1990] James Hendler, Austin Tate, and Mark Drummond. AI planning: Systems and techniques. *AI Magazine*, 11(2):61–77, Summer 1990.

[Horvitz *et al.*, 1989] Eric J. Horvitz, Gregory F. Cooper, and David E. Heckerman. Reflection and action under scarce resources: theoretical principles and empirical study. In *Proceedings of IJCAI-89*, pages 1121–1127, Detroit, MI, August 1989.

[Kaelbling, 1988] Leslie Pack Kaelbling. Goals as parallel program specifications. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 60–65, St. Paul, MN, August 1988.

[Krebsbach *et al.*, 1991] Kurt Krebsbach, Duane Olawsky, and Maria Gini. Deferring task planning in the tool box world: Empirical results. Technical Report TR 91-60, University of Minnesota Department of Computer Science, Minneapolis, MN, 1991.

[Krebsbach *et al.*, 1992] Kurt Krebsbach, Duane Olawsky, and Maria Gini. Sensing and deferral in planning: Empirical results. In *Proceedings of the First International Conference on AI Planning Systems*, pages 136–144, College Park, Maryland, June 1992.

[Krebsbach, 1993] Kurt Krebsbach. Rational sensing for an AI planner: A cost-based approach. Ph.d. dissertation, University of Minnesota Department of Computer Science, 1993.

[McDermott, 1978] Drew McDermott. Planning and acting. *Cognitive Science*, 2:71–109, 1978.

[McDermott, 1992] Drew McDermott. Robot planning. *AI Magazine*, pages 55–79, Summer 1992.

[Moore, 1985] Robert L. Moore. A formal theory of knowledge and action. In *Formal Theories of the Commonsense World*, pages 319–358. Ablex Publishing Corporation, 1985.

[Olawsky and Gini, 1990] Duane Olawsky and Maria Gini. Deferred planning and sensor use. In *Innovative Approaches to Planning, Scheduling and Control: Proceedings of the DARPA Workshop on Planning*, pages 166–174, San Diego, CA, November 1990.

[Pednault, 1989] E. P. D. Pednault. ADL: Exploring the middle ground beween STRIPS and the situation calculus. In *International Conference on Knowledge Representation and Reasoning*, 1989.

[Peot and Smith, 1992] Mark Peot and David Smith. Conditional nonlinear planning. In *Proceedings of the First International Conference on AI Planning Systems*, pages 189–197, College Park, Maryland, June 1992.

[Schoppers, 1987] Marcel Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1039–1046, Milan, Italy, August 1987.

[Turney and Segre, 1989] Jennifer Turney and Alberto Segre. A framework for learning in planning domains with uncertainty. Technical Report 89-1009, Department of Computer Science, Cornell University, Ithaca, NY, May 1989.

[Wellman and Doyle, 1992] M. Wellman and J. Doyle. Modular utility representation for decision-theoretic planning. In *International Conference on AI Planning Systems*, pages 236–242, 1992.

[Wilkins *et al.*, 1994] D. E Wilkins, K. L. Myers, J. D. Lowrance, and L. P. Wesley. Planning and reacting in uncertain and dynamic environments. *Journal of Experimental and Theoretical Artificial Intelligence*, 6:197–227, 1994.