# Probabilistic Planning for Behavior-Based Robots*

**Amin Atrash** and **Sven Koenig**
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
{amin, skoenig}@cc.gatech.edu

## Abstract

Partially Observable Markov Decision Process models (POMDPs) have been applied to low-level robot control. We show how to use POMDPs differently, namely for sensor-planning in the context of behavior-based robot systems. This is possible because solutions of POMDPs can be expressed as policy graphs, which are similar to the finite state automata that behavior-based systems use to sequence their behaviors. An advantage of our system over previous POMDP navigation systems is that it is able to find close-to-optimal plans since it plans at a higher level and thus with smaller state spaces. An advantage of our system over behavior-based systems that need to get programmed by their users is that it can optimize plans during missions and thus deal robustly with probabilistic models that are initially inaccurate.

## Introduction

Mobile robots have to deal with various kinds of uncertainty, such as noisy actuators, noisy sensors, and uncertainty about the environment. Behavior-based robot systems, such as MissionLab (Endo *et al.* 2000), can operate robustly in the presence of uncertainty (Arkin 1998). Its operation is controlled by plans in form of finite state automata, whose states correspond to behaviors and whose arcs correspond to observations. These finite state automata have to be programmed by the users of the system at the beginning of a mission. However, plans generated by humans are rarely optimal because they involve complex tradeoffs. Consider, for example, a simple sensor-planning task, where a robot has to decide how often to sense before it starts to act. Since the sensors of the robot are noisy, it may have to sense multiple times. On the other hand, sensing takes time. How often the robot should sense depends on the amount of sensor noise, the cost of sensing, and the consequences of acting based on wrong sensor information.

In this paper, we develop a robot architecture that uses Partially Observable Markov Decision Process models (POMDPs) (Sondik 1978) for planning and combines them with MissionLab. POMDPs provide an elegant and theoretically grounded way for probabilistic planning (Cassandra, Kaelbling, & Littman 1994). So far, they have been used mainly to solve low-level planning tasks for mobile robots such as path following and localization (Fox, Burgard, & Thrun 1998; Mahadevan, Theocharous, & Khaleeli 1998; Cassandra, Kaelbling, & Kurien 1996; Simmons & Koenig 1995). In this paper, we show that POMDPs can also be used to solve higher-level planning tasks for mobile robots. The key idea behind our robot architecture is that POMDP planners can generate policy graphs rather than the more popular value surfaces. Policy graphs are similar to the finite state automata of MissionLab. An advantage of our robot architecture is that it uses POMDPs in small state spaces. When POMDPs are used for low-level planning, the state spaces are often large and finding optimal or close-to-optimal POMDPs becomes extremely time-consuming (Papadimitriou & Tsitsiklis 1987). Thus, existing robot systems have so far only been able to use greedy POMDP planning methods that produce extremely suboptimal plans (Koenig & Simmons 1998). Our robot architecture, on the other hand, is able to find close-to-optimal plans.

In the following, we first give an example of sensor planning and then give overviews of behavior-based robotics and POMDPs using this example. Next, we describe how our robot architecture combines these ideas by transforming the output of the POMDP planner (policy graphs) to the input of MissionLab (finite state automata). Finally, we report on two experiments that show that the ability to optimize plans during missions is important because the resulting system is able to deal robustly with probabilistic models that are initially inaccurate.

## Example: Sensor Planning

We use the following sensor-planning example throughout this paper, which is similar to an example used in (Cassandra, Kaelbling, & Littman 1994). Assume that a police robot attempts to find wounded hostages in a building. When it is at a doorway, it has to decide whether to search the room. The robot can either use its microphone to listen for terrorists (OBSERVE); enter the room, look around, leave the room, and proceed to the next doorway (ENTER ROOM AND PROCEED); or move to the next doorway right away

*This research is supported by DARPA/U.S. Army SMDC contract #DASG60-99-C-0081. Approved for public release; distribution unlimited. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations and agencies or the U.S. government.

(PROCEED). The cost of OBSERVE is always 5, and the cost of PROCEED is always 50. Each room is occupied by terrorists with probability 0.5. OBSERVE reports either that the room is occupied by terrorists (OBSERVE OCCUPIED) or not (OBSERVE EMPTY). Although the microphone always detects the absence of terrorists, it does not detect the presence of terrorists with probability 0.2. Multiple observations are drawn independently from this probability distribution, which is not completely unrealistic for sound. The robot gets a reward for entering a room, as an incentive to find wounded hostages. However, it also gets a penalty if the room is occupied by terrorists since terrorists might destroy it. If the room is not occupied by terrorists (ROOM EMPTY), then ENTER ROOM AND PROCEED results in a reward of 100. However, if the room is occupied by terrorists (ROOM OCCUPIED), then ENTER ROOM AND PROCEED results in a penalty of 500. The main decision that the robot has to make is how often to OBSERVE and, depending on the sensor observations, whether to PROCEED to the next doorway right away or to first ENTER the ROOM AND then PROCEED.

## Behavior-Based Robotics

Behavior-based robotics uses a tight coupling between sensing and acting to operate robustly in the presence of uncertainty. The robot always executes a behavior such as "move to the doorway" or "enter the room." To sequence these behaviors, behavior-based robotics often uses finite state automata whose states correspond to behaviors and whose arcs correspond to triggers (observations). The current state dictates the behavior of the robot. When an observation is made and there is an edge labeled with this observation that leaves the current state, the current state changes to the state pointed to by the edge. Since the finite state automata are based on behaviors and triggers, the robot does not require a model of the world or complete information about the current state of the world. For example, a robot does not need to know the number of doorways or the distances between them.

We use a robot system based on MissionLab (Endo *et al.* 2000). MissionLab provides a large number of behaviors and triggers with which users can build finite state automata, that can then be executed on a variety of robots or in simulation. The finite state automata have to be programmed by the users of the system at the beginning of a mission. This has the disadvantage that MissionLab cannot optimize the finite state automata during the mission, for example, when it learns more accurate probabilities or when the environments change. Furthermore, humans often assume that sensor are accurate. Their plans are therefore often suboptimal. We address this issue by developing a robot architecture which uses a POMDP planner to generate plans based on probabilistic models of the world.

## POMDPs

POMDPs consist of a finite set of states $S$, a finite set of observations $O$, and an initial state distribution $\pi$. Each state $s \in S$ has a finite set of actions $A(s)$ that can be executed in it. The POMDP further consists of a transition function $p$, where $p(s'|s,a)$ denotes the probability with which the system transitions from state $s$ to state $s'$ when action $a$ is executed, an observation function $q$, where $q(o|s,a)$ denotes the probability of making observation $o$ when action $a$ is executed in state $s$, and a reward function $r$, where $r(s,a)$ denotes the finite reward (negative cost) that results when action $a$ is executed in state $s$. A POMDP process is a stream of <state, observation, action, reward> quadruples. The POMDP process is always in exactly one state and makes state transitions at discrete time steps. The initial state of the POMDP process is drawn according to the probabilities $\pi(s)$. Thus, $p(s_t = s) = \pi(s)$ for $t = 1$. Assume that at time $t$, the POMDP process is in state $s_t \in S$. Then, a decision maker chooses an action $a_t$ from $A(s_t)$ for execution. This results in reward $r_t = r(s_t, a_t)$ and observation $o_t \in O$ that is generated according to the probabilities $p(o_t = o) = q(o|s_t, a_t)$. Next, the POMDP process changes state. The successor state $s_{t+1} \in S$ is selected according to the probabilities $p(s_{t+1} = s) = p(s|s_t, a_t)$. This process repeats forever.

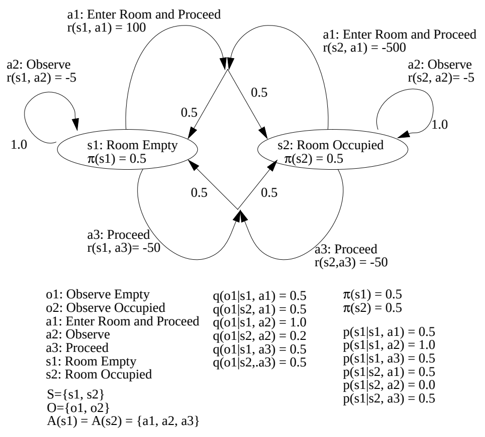

Figure 1: POMDP

As an example, Figure 1 shows the POMDP that corresponds to our sensor-planning task. The robot starts at a doorway without knowing whether the room is occupied. Thus, it is in state ROOM OCCUPIED with probability 0.5 and ROOM EMPTY with probability 0.5 but does not know which one it is in. In both states, the robot can OBSERVE, PROCEED to the next doorway, or ENTER ROOM AND PROCEED. OBSERVE does not change the state but the sensor observation provides information about it. PROCEED and ENTER ROOM AND PROCEED both result in the robot being at the next doorway and thus again in state ROOM OCCUPIED with probability 0.5 and ROOM EMPTY with probability 0.5. The observation probabilities and rewards of the actions are as described above.

## Policy Graphs

Assume that a decision maker has to determine which action to execute for a given POMDP at time $t$. The decision maker knows the specification of the POMDP, executed the

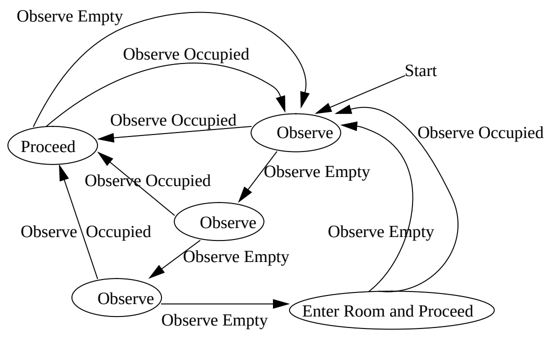

Figure 2: Policy Graph

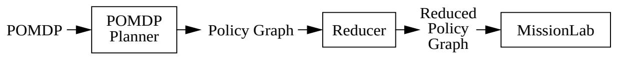

Figure 3: Robot Architecture

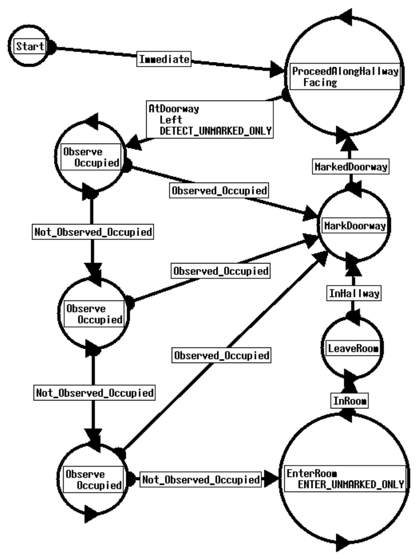

Figure 4: Finite State Automaton

actions $a_1 \dots a_{t-1}$, and made the observations $o_1 \dots o_{t-1}$. The objective of the decision maker is to maximize the average total reward over an infinite planning horizon, which is $E(\sum_{t=1}^{\infty}[\gamma^{t-1}r_t])$, where $\gamma \in (0,1]$ is a discount factor. The discount factor specifies the relative value of a reward received after $t$ action executions compared to the same reward received one action execution earlier. One often uses a discount factor slightly smaller than one because this ensures that the average total reward is finite, no matter which actions are chosen. (We use $\gamma = 0.99$). In our case, the robot is the decision maker who executes movement and sensing actions and receives information about the state of the world from inaccurate sensors, such as the microphone. We let the robot maximize the average total reward over an infinite horizon because it searches a large number of rooms.

It is a fundamental result of operations research that optimal behaviors for the robot can be expressed either as value surfaces or policy graphs (Sondik 1978). Value surfaces are mappings from probability distributions over the states to values. The robot calculates the expected value of the probability distribution over the states that results from the execution of each action and then chooses the action that results in the largest expected value. Policy graphs are graphs where the vertices correspond to actions and the directed edges correspond to observations. The robot executes the action that corresponds to its current vertex. Then, it makes an observation, follows the corresponding edge, and repeats the process.

It is far more common for POMDP planners to use value surfaces than policy graphs. However, policy graphs allow us to integrate POMDP planning and behavior-based systems because of their similarity to finite state automata. As an example, Figure 2 shows the optimal policy graph for our sensor-planning task. This policy graph specifies a behavior where the robot senses three times before it decides to enter a room. If any of the sensing operations indicates that the room is occupied, the robot decides to move to the next doorway without entering the room.

Optimal policy graphs can potentially be large but often turn out to be very small (Cassandra, Kaelbling, & Littman 1994). However, finding optimal or close-to-optimal policy graphs is PSPACE-complete in general (Papadimitriou & Tsitsiklis 1987) and thus only feasible for small planning tasks. We decided to use a POMDP planner that was developed by Hansen in his dissertation at the University of Massachusetts at Amherst (Hansen 1998). This POMDP planner can often find optimal or close-to-optimal policy graphs for our POMDP problems in seconds. We use the POMDP planner unchanged, with one small exception. We noticed that many vertices of the policy graph are often unreachable from the start vertex and eliminate these vertices using a simple graph-search technique.

## The Robot Architecture

Figure 3 shows a flow graph of our robot architecture. The user inputs a POMDP that models the planning task. The robot architecture then uses the POMDP planner to produce a policy graph and removes all vertices that are unreachable from the initial vertex. By mapping the actions of the policy graph to behaviors and the observations to triggers, the policy graph is then transformed to a finite state automaton and used to control the operation of MissionLab. The user still has to input information but now only the planning task and its parameters (that is, the probabilities and costs) and no longer the plans. Once the finite state automaton is read into MissionLab, we allow the user to examine and edit it, for example, to add additional parts to the mission or make it part of a larger finite state automaton. Figure 4 shows a screenshot of the policy graph from Figure 2 after it was read into MissionLab and augmented with details about how to implement PROCEED (namely by marking the current doorway and proceeding along the hallway until the robot is at an unmarked doorway) and ENTER ROOM AND PROCEED (namely by entering the room, leaving the room, marking the current doorway, and proceeding along the hallway until the robot is at an unmarked doorway). Furthermore, the user decided that it was more robust to start with the behavior that proceeds along the hallway because then the robot can start anywhere in the hallway and not only at doorways.

Our robot architecture shows that it is possible to integrate POMDP planning and behavior-based systems, by specifying the solution of POMDPs in form of policy graphs. However, there are small semantic differences between policy graphs and finite state automata. POMDPs assume that actions are discrete and that the robot makes an observation after each action execution. Finite state automata assume that behaviors are continuous and triggers can be observed at any time during the execution of the behaviors. Two issues need to be addressed in this context. First, we need MissionLab to be able to deal with actions of finite duration. We deal with this problem by adding an ACTION FINISHED trigger to MissionLab. (This extension is not needed for our sensing-planning task.) Second, we need to deal with a potential combinatorial explosion of the number of observations. Most POMDP planners assume that every observation can be made in every state. Consequently, every vertex in a policy graph has one outgoing edge for each possible observation. However, the observations are $n$ tuples if there are $n$ sensors and the number of observations can thus be large. This is not a problem for finite state automata since observations that do not cause state transitions do not appear in them. We deal with this problem by omitting subtasks from the POMDP planning task that can be abstracted away or are pre-sequenced and do not need to be planned. For example, ENTER ROOM AND PROCEED is a macro-behavior that consists of a sequence of observations and behaviors, as shown in Figure 4. By omitting the details of ENTER ROOM AND PROCEED, the observations IN ROOM, IN HALLWAY, and MARKED DOORWAY do not need to be considered during planning.

## Experiments

We test the performance of our system, both analytically and experimentally, by comparing the average total reward of its plans (that is, optimal plans) against the plans typically generated by users. For our sensor-planning task, users typically create plans that sense only once, no matter what the probabilities and costs are. The robot executes ENTER ROOM AND PROCEED if it senses ROOM EMPTY, otherwise it executes PROCEED. We therefore use this plan as baseline plan and compare the plans generated by our system against it.

**Analytical Results:** To demonstrate that our system has an advantage over the previous system because it is able to optimize its plans during missions when it is able to estimate the costs more precisely, we determine analytically how the average total reward of the baseline plan depends on the reward $x = r(s_2, a_1)$ for entering an occupied room. Let $k$ be the state directly before the robot executes OBSERVE, $l$ the state directly before it executes ENTER ROOM AND PROCEED, and $m$ the state directly before it executes PROCEED. If the robot is in $k$, then it incurs a cost of 5 for executing OBSERVE. It then transitions from $k$ to $l$ with the probability with which the sensor reports OBSERVE EMPTY, otherwise it transitions to $m$.

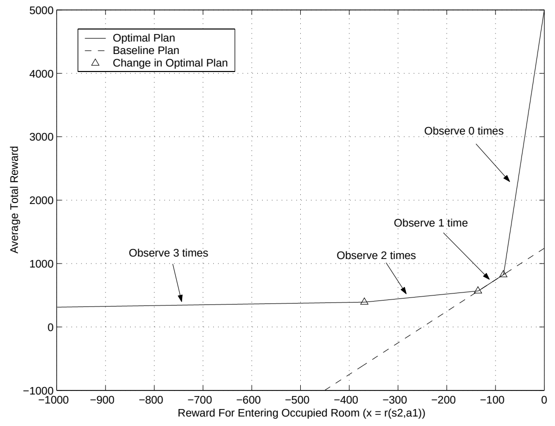

Figure 5: Average Total Rewards vs. Reward for Entering an Occupied Room (x) (Analytical)

Using the notation of Figure 1, the probability $p(o_t = o_1)$ with which the sensor reports OBSERVE EMPTY is $p(o_t = o_1) = q(o_1|s_1, a_2)p(s_t = s_1) + q(o_1|s_2, a_2)p(s_t = s_2) = 1.0\,0.5 + 0.2\,0.5 = 3/5$. Consequently, the average total reward $v(k)$ of the baseline plan if the robot starts in $k$ is $v(k) = -5 + \gamma(3/5v(l) + 2/5v(m))$. Similar derivations result in a system of three linear equations in three unknowns:

$$\begin{aligned} v(k) &= -5 + \gamma(3/5v(l) + 2/5v(m)) \\ v(l) &= 1/6x + 5/6\,100 + \gamma v(k) \\ v(m) &= -50 + \gamma v(k) \end{aligned}$$

Solving this system of equations yields $v(k) = 1241.21 + 4.97x$. Figure 5 shows this graph together with the average total reward of the plans generated by our system, as a function of $x$. As can be seen, the number of times a robot has to sense OBSERVE EMPTY before it enters a room increases as it becomes more expensive to enter an occupied room. (The markers show when a change in plan occurs.) The robot pays a cost for the additional sensing operations but this decreases the probability of entering an occupied room. Changing the plans as $x$ changes allows the average total reward of the plans generated by our system to deteriorate much more slowly than the average total reward of the baseline plan. This result shows that our system has an advantage over the previous system because it is able to adapt plans during missions when the costs can be estimated more precisely. It also shows that our system has an advantage over the previous system because humans are not good at planning with uncertainty and thus their plans are rarely optimal. For example, the original sensor-planning problem has $x = -500$ and the average total reward of the baseline plan is only -1,246.24 whereas the average total reward of the plan generated by our system is 374.98.

Similar results can be observed if the initial probabilities are inaccurate. Figure 6 shows the average total reward of the baseline plan together with the average total reward of the plans generated by our system, as a function of the probability $y = q(o_2|s_2, a_2)$ with which the microphone correctly classifies an occupied room, for both $x = -200$ and $x = -500$. As can be seen, the number of times a robot

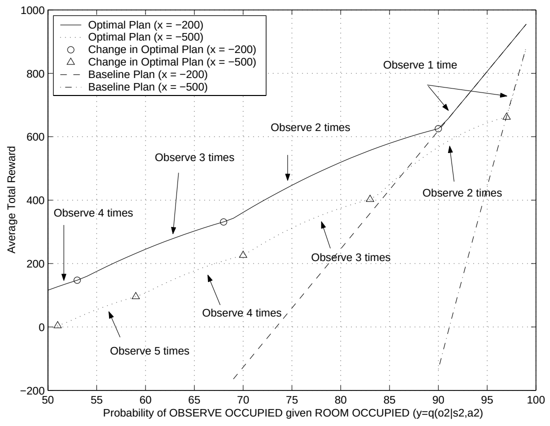

Figure 6: Average Total Rewards vs. Probability of Correctly Classifying an Occupied Room (Analytical)

| | Baseline Plan | Optimal Plan |
|---|---|---|
| Reward for Entering Occupied Room x = -200 | 16.92 | 282.89 |
| Reward for Entering Occupied Room x = -500 | -1000.42 | 498.23 |

Figure 7: Average Total Rewards (Experimental)

has to sense OBSERVE EMPTY before it enters a room increases as the sensor becomes more noisy. This again allows the average total reward of the plans generated by our system to deteriorate much more slowly than the average total reward of the baseline plan, demonstrating the advantages of our system.

**Experimental Results:** We also performed a simulation study with MissionLab to compare the average total reward of the plans generated by our system against the baseline plan, for $y = 0.8$ and both $x = -200$ and $x = -500$. We used four rooms and averaged over ten runs. Figure 7 shows the results. In both cases, the average total reward of the baseline plan is much smaller than the average total reward of the plans generated by our system. (The table shows that the average total reward of the plans generated by our system actually increased as it became more costly to enter an occupied room. This artifact is due to the reduced probability of entering an occupied room, causing the situation to never occur during our limited number of runs.) These results are similar to the analytical results shown in Figure 5.

## Conclusion

This paper reported on initial work that uses Partially Observable Markov Decision Process models (POMDPs) in the context of behavior-based systems. The insight to making this combination work is that POMDP planners can generate policy graphs rather than the more popular value surfaces, and policy graphs are similar to the finite state automata that behavior-based systems use to sequence their behaviors. This combination also keeps the POMDPs small, which allows our POMDP planners to find optimal or close-to-optimal plans whereas the POMDP planners of other robot architectures can only find very suboptimal plans.

We used this insight to improve MissionLab, a behavior-based system where the finite state automata had to be programmed by the users of the system at the beginning of the mission. This had the disadvantage that humans are not good at planning with uncertainty and thus their plans are rarely optimal. In contrast, our robot architecture does not only produce close-to-optimal plans but is also able to optimize the finite state automata when it learns more accurate probabilities or when the environment changes.

It is future work to study interfaces that allow users to easily input POMDPs, including probabilities and costs. Also, we intend to implement sampling methods for adapting the probabilities and costs of POMDPs during missions to be able to update the plan during execution. Finally, it is future work to scale up our robot architecture by developing POMDP planners that are able to take advantage of the structure of the POMDP planning tasks and thus are more efficient than current POMDP planners.

## References

Arkin, R. 1998. *Behavior-Based Robotics*. MIT Press.

Cassandra, A.; Kaelbling, L.; and Kurien, J. 1996. Acting under uncertainty: Discrete Bayesian models for mobile robot navigation. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 963–972.

Cassandra, A.; Kaelbling, L.; and Littman, M. 1994. Acting optimally in partially observable stochastic domains. In *Proceedings of the National Conference on Artificial Intelligence*, 1023–1028.

Endo, Y.; MacKenzie, D.; Stoychev, A.; Halliburton, W.; Ali, K.; Balch, T.; Cameron, J.; and Chen, Z. 2000. MissionLab: User manual for MissionLab version 4.0. Technical report, College of Computing, Georgia Institute of Technology.

Fox, D.; Burgard, W.; and Thrun, S. 1998. Active markov localization for mobile robots. *Robotics and Autonomous Systems* 25:195–207.

Hansen, E. 1998. Solving POMDPs by searching in policy space. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 211–219.

Koenig, S., and Simmons, R. 1998. Xavier: A robot navigation architecture based on partially observable Markov decision process models. In Kortenkamp, D.; Bonasso, R.; and Murphy, R., eds., *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*. MIT Press. 91–122.

Mahadevan, S.; Theocharous, G.; and Khaleeli, N. 1998. Rapid concept learning for mobile robots. *Autonomous Robots Journal* 5:239–251.

Papadimitriou, C., and Tsitsiklis, J. 1987. The complexity of Markov decision processes. *Mathematics of Operations Research* 12(3):441–450.

Puterman, M. 1994. *Markov Decision Processes – Discrete Stochastic Dynamic Programming*. Wiley.

Simmons, R., and Koenig, S. 1995. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1080–1087.

Sondik, E. 1978. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research* 26(2):282–304.