Journal of Artificial Intelligence Research 42 (2011) 815-850 Submitted 07/11; published 12/11

Stochastic Enforced Hill-Climbing

Jia-Hong Wu JW@ALUMNI.PURDUE.EDU
Institute of Statistical Science,
Academia Sinica, Taipei 115, Taiwan ROC
Rajesh Kalyanam RKALYANA@PURDUE.EDU
Robert Givan GIVAN@PURDUE.EDU
Electrical and Computer Engineering,
Purdue University, W. Lafayette, IN 47907, USA

Abstract

Enforced hill-climbing is an effective deterministic hill-climbing technique that deals with local optima using breadth-first search (a process called “basin flooding”). We propose and evaluate a stochastic generalization of enforced hill-climbing for online use in goal-oriented probabilistic planning problems. We assume a provided heuristic function estimating expected cost to the goal with flaws such as local optima and plateaus that thwart straightforward greedy action choice. While breadth-first search is effective in exploring basins around local optima in deterministic problems, for stochastic problems we dynamically build and solve a heuristic-based Markov decision process (MDP) model of the basin in order to find a good escape policy exiting the local optimum. We note that building this model involves integrating the heuristic into the MDP problem because the local goal is to improve the heuristic.

We evaluate our proposal in twenty-four recent probabilistic planning-competition benchmark domains and twelve probabilistically interesting problems from recent literature. For evaluation, we show that stochastic enforced hill-climbing (SEH) produces better policies than greedy heuristic following for value/cost functions derived in two very different ways: one type derived by using deterministic heuristics on a deterministic relaxation and a second type derived by automatic learning of Bellman-error features from domain-specific experience. Using the first type of heuristic, SEH is shown to generally outperform all planners from the first three international probabilistic planning competitions.

1. Introduction

Heuristic estimates of distance-to-the-goal have long been used in deterministic search and deterministic planning. Such estimates typically have flaws such as local extrema and plateaus that limit their utility. Methods such as simulated annealing (Kirkpatrick, Gelatt, & Vecchi, 1983; Cerny, 1985) and A* (Nilsson, 1980) search have been developed for handling flaws in heuristics. More recently, excellent practical results have been obtained by “flooding” local optima using breadth-first search. This method is called “enforced hill-climbing” (Hoffmann & Nebel, 2001).

Deterministic enforced hill-climbing (DEH) is proposed in the work of Hoffmann and Nebel (2001) as a core element of the successful deterministic planner Fast-Forward (FF). DEH is an extension of the basic “hill-climbing” approach of simply selecting actions greedily by looking ahead one action step, and terminating when reaching a local optimum. DEH extends basic hill-climbing by replacing termination at local optima with breadth-first search to find a successor state with strictly better heuristic value. The planner then moves to that descendant and repeats this process. DEH is guaranteed to find a path to the goal if the problem itself is deadend-free (so that every state has such a path). While that relatively weak guarantee applies independent of the quality of the heuristic function, the intent of DEH is to remediate flaws in a generally accurate heuristic in order to leverage that heuristic in finding short paths to the goal. In domains where the basin size (search depth needed to escape any optimum) is bounded, DEH can provide a polynomial-time solution method (Hoffmann, 2005).

Enforced hill-climbing is not defined for probabilistic problems, due to the stochastic outcomes of actions. In the presence of stochastic outcomes, finding descendants of better values no longer implies the existence of a policy that reaches those descendants with high probability. One may argue that FF-Replan (Yoon, Fern, & Givan, 2007)—a top performer for recent probabilistic planning benchmarks—uses enforced hill-climbing during its call to FF. However, the enforced hill-climbing process is used on a determinized problem, and FF-Replan does not use any form of hill climbing directly in the stochastic problem. In fact, FF-Replan does not consider the outcome probabilities at all.

One problem to consider in generalizing enforced hill-climbing to stochastic domains is that the solution to a deterministic problem is typically concise, a sequential plan. In contrast, the solution to a stochastic problem is a policy (action choice) for all possibly reached states. The essential motivation for hill-climbing is to avoid storing exponential information during search, and even the explicit solution to a stochastic problem cannot be directly stored while respecting this motivation. For this reason, we limit consideration to the online setting, where the solution to the problem is a local policy around the current state. After this local policy is committed to and executed until the local region is exited, the planner then has a new online problem to solve (possibly retaining some information from the previous solution). Our approach generalizes directly to the construction of offline policies in situations where space to store such policies is available. Note that, in contrast, deterministic enforced hill-climbing is easily implemented as an offline solution technique.

We propose a novel tool for stochastic planning by generalizing enforced hill-climbing to goal-based stochastic domains. Rather than seeking a sequence of actions deterministically leading to a better state, our method uses a finite-horizon MDP analysis around the current state to seek a *policy* that expects to improve on the heuristic value of the current state. Critical to this process is the direct incorporation of both the probabilistic model and the heuristic function in finding the desired policy. Therefore, for the finite-horizon analysis, the heuristic function is integrated into the MDP problem in order to represent the temporary, greedy goal of improving on the current heuristic value. This integration is done by building a novel "heuristic-based MDP" in which any state has a new "exit" action available that terminates execution with cost equal to the heuristic estimate for that state, and all other action costs are removed[1]. In a heuristic-based MDP, finite-horizon policies are restricted by a requirement that at horizon one, the exit action must be selected (but can also be selected at other horizons). In this heuristic-based MDP, the cost of any policy π at a state s is the expected value of the heuristic upon exit (or horizon) if π is executed from s.

Thus, we find the desired local policy using value iteration on the heuristic-based MDP around the current state, with deepening horizon, until a policy is found with cost improving on the heuristic estimate at the current state. The restriction of selecting the exit action at horizon one corresponds to initializing value iteration with the provided heuristic function. When such a policy is found, the

[1] The motivation for the removal of action costs in the heuristic-based MDP is discussed in Section 3.2.

method executes the policy until an exiting action is indicated (or to the horizon used in computing the policy).

The resulting method, stochastic enforced hill-climbing (SEH), simply generalizes depth-k breadth-first search for a state with improved heuristic value (from DEH) to a k-horizon value iteration computation seeking a policy that expects improvement in heuristic value. Note that although stochastic enforced hill-climbing is an explicit statespace technique, it can be suitable for use in astronomically large statespaces if the heuristic used is informative enough to limit the effective size of the horizon k needed to find expected heuristic improvement. Our empirical results in this work demonstrate this behavior successfully.

1.1 Applicability and Limitations

Stochastic enforced hill-climbing (SEH) can be applied to any heuristic function. However, the applicability (and likewise the limitations) of SEH greatly depends on the characteristics of the heuristic function. SEH is appropriate in any goal-oriented problem given a strong enough heuristic function, and we demonstrate empirically that SEH generally outperforms greedy following of the same heuristic for a variety of heuristics in a variety of domains, even in presence of probabilistically interesting features (Little & Thiebaux, 2007) and deadends. SEH can rely upon the heuristic function for identification of dead-ends and appropriate handling of probabilistically interesting features that require non-local analysis—SEH simply provides local search that often can correct other flaws in the heuristic function. SEH is thus intended as a possible improvement over stochastic solution methods that construct a cost-to-go (cost) function and follow it greedily when using the constructed cost function as a search heuristic. Many methods for constructing value/cost functions have been proposed and evaluated in the literature, all of which can potentially be improved for goal-based domains by using SEH in place of greedy following (Sutton, 1988; Fahlman & Lebiere, 1990; Bertsekas, 1995; Gordon, 1995; Mahadevan & Maggioni, 2007; Sanner & Boutilier, 2009)[2]. We prove the correctness of SEH in Section 3.4 by showing that in deadend-free domains, SEH finds the goal with probability one (i.e. SEH does not get stuck in local optima).

While SEH is a search technique that leverages a heuristic estimate of distance to go, it must be emphasized that, unlike many other such search techniques, SEH makes no promises about the optimality of the solution path found. SEH is a greedy, local technique and can only promise to repeatedly find a policy that reduces the heuristic value, and only when that is possible. As such, SEH is an inappropriate technique for use when optimal solutions are required.

Stochastic enforced hill-climbing can be ineffective in the presence of huge plateaus or valleys in the heuristic functions, due to extreme resource consumption in finding desired local policies. Heuristic functions with huge plateaus result from methods that have failed to find any useful information about the problem in those state regions. SEH is inappropriate as the only tool for solving a stochastic planning problem—other tools are needed to construct a useful heuristic function that manages deadends and avoids huge plateaus. This weakness mirrors the weakness of enforced hill-climbing in deterministic domains. SEH can also fail to find the goals when avoidable dead-ends are present but not recognized early enough by the heuristic. In fact, effective dead-end detection is a central goal in heuristic design when any greedy technique will be applied to the heuristic.

[2] For applicability of SEH, a cost function must be non-negative and must identify goals by assigning zero to a state if and only if it is a goal state; however, more general value/cost functions can be normalized to satisfy these requirements.

Further insight into the usefulness of SEH can be gained by comparison with recent determinizing replanners. As mentioned above, one way to exploit deterministic planning techniques such as DEH for stochastic problems is to determinize the planning problem and use a deterministic planner to select an action sequence. Executing this action sequence in the problem is not guaranteed to reach the goal due to the determinization approximation, so replanning is needed to augment this technique. In this paper, we call stochastic planners that use this technique "determinizing replanners". Determinizing replanners using a determinization (called "all outcomes") that retains all possible state transitions can be shown to reach the goal with probability one in the absence of dead-end states.

In contrast to determinizing replanners, SEH at no point relies on any determinization of the problem, but instead analyzes increasing-size local probabilistic approximations to the problem. SEH conducts a full probabilistic analysis within the horizon, seeking the objective of reducing the provided heuristic, using value iteration. In this way, SEH leverages the probabilistic parameters that are ignored by determinizing replanners, as well as the provided heuristic function, which can be based upon substantial probabilistic analysis. As a result, SEH successfully handles probabilistic problem aspects that cause major problems for determinizing replanners. However, at this point, we have no theoretical results characterizing its gains over determinizing replanners. Instead, we have an extensive empirical evaluation showing advantages over FF-Replan (Yoon et al., 2007) and RFF (Teichteil-Konigsbuch, Kuter, & Infantes, 2010) (two determinizing replanners), as well as substantial gains compared to greedy following of the heuristic (which also uses the transition probability parameters).

1.2 Evaluation

We test SEH on a broad range of domains from the first three international probabilistic planning competitions (as well as the "probabilistically interesting" domains in Little & Thiebaux, 2007), using two very different methods to generate heuristic functions. First, we test SEH on a heuristic function based on the ideas of the successful re-planner FF-Replan (Yoon et al., 2007). This new "controlled-randomness FF (CR-FF) heuristic" is the deterministic FF heuristic (Hoffmann & Nebel, 2001) computed on the simple determinization of the probabilistic problem that makes available a deterministic transition wherever a probabilistic transition was possible. We note that FF-Replan itself does not use this (or any) heuristic function in the stochastic problem. Instead, FF-Replan relies on FF to construct a plan in the deterministic problem, and these calls to FF in turn use deterministic enforced hill-climbing with exactly this heuristic. Here, we consider the performance of this heuristic directly in the stochastic problem, comparing greedy heuristic-following with SEH-based search around the heuristic. The latter method using SEH constitutes a novel method for combining determinization (that removes the probabilistic parameters) with probabilistic reasoning. Our experiments show that this new method substantially outperforms FF-Replan across our broad evaluation.

We have also performed a second evaluation of our technique on heuristic functions learned from domain-specific experience by the relational feature-learning method presented in the work of Wu and Givan (2007, 2010). These heuristic functions have already been shown to give good performance when used to construct a simple greedy policy, and are further improved by SEH.

The SEH technique can be seen to perform well in a domain-by-domain analysis across the broad set of competition planning domains, and full domain-by-domain results are available in an online appendix. However, to compress and summarize the extensive per-problem results, we have divided all the evaluation domains into experimenter-defined “categories” and have aggregated performance measurement within each problem category. While some categories are single domains, more generally, multiple closely related domains may be aggregated within a single category. For example, multiple domains from the competitions have been variants of the “blocks world”, and problems in these domains are aggregated as a BLOCKSWORLD category.

In order to fairly compare SEH with FF-based planners (such as RFF, as described in Teichteil-Konigsbuch et al., 2010, and FF-Replan) that exploit the blocksworld-targeted planning heuristics “added goal deletion” and “goal agenda”, we have provided these heuristics as extensions to SEH. The resulting planner is called SEH$^+$, described in detail in Section 3.6. Our results show that SEH$^+$ performs nearly identically to SEH on non-blocksworld categories when using the CR-FF heuristic. We employ these extensions when comparing SEH with the CR-FF heuristic to other planners.

Using experimenter-defined categories, we are able to show that SEH exploits the heuristic functions more effectively than greedy following of the heuristic. SEH statistically significantly outperforms greedy following in thirteen out of seventeen categories using the CR-FF heuristics while losing in one category. SEH also outperforms greedy following in six out of seven categories using the learned heuristics. (In both cases, the other categories showed similar performance between the compared planners.)

We show that SEH$^+$, when using the CR-FF heuristics, outperforms FF-Replan on ten out of fifteen categories, with similar performance on two more categories, losing on only three categories. Our aggregate results show that SEH$^+$ (using the CR-FF heuristics) has a particularly strong performance advantage over FF-Replan in “probabilistically interesting” categories (Little & Thiebaux, 2007).

Finally, we compare the performance of SEH$^+$ against that of RFF-BG (Teichteil-Konigsbuch et al., 2010), one winner of the fully-observable track of the third international probabilistic planning competition. SEH$^+$ outperforms RFF-BG on twelve out of fifteen categories, with similar performance on one more category, losing on only two categories.

In summary, our empirical work demonstrates that SEH provides a novel automatic technique for improving on a heuristic function using limited searches, and that simply applying SEH to reasonable heuristic functions produces a state-of-the-art planner.

2. Technical Background: Markov Decision Processes

We give a brief review of Markov decision processes (MDPs) specialized to goal-region objectives. For more detail on MDPs, see the work of Bertsekas (1995), Puterman (2005), and Sutton and Barto (1998).

2.1 Goal-Oriented Markov Decision Processes

A Markov decision process (MDP) M is a tuple $(S, A, C, T, s_{\text{init}})$. Here, S is a finite state space containing initial state s_{init}, and A selects a non-empty finite available action set $A(s)$ for each state s in S. The action-cost function C assigns a non-negative real action-cost to each state-action-state triple (s, a, s') where action a is enabled in state s, i.e., a is in $A(s)$. The transition probability function T maps state-action pairs (s, a) to probability distributions over S, $\mathcal{P}(S)$, where a is in $A(s)$.

To represent the goal, we include in S a zero-cost absorbing state $\perp$, i.e., such that $C(\perp, a, s) = 0$ and $T(\perp, a, \perp) = 1$ for all $s \in S$ and $a \in A(\perp)$. Goal-oriented MDPs are MDPs where there is a subset $G \subseteq S$ of the statespace, containing $\perp$, such that: (1) $C(g, a, s')$ is zero whenever $g \in G$ and one otherwise, and (2) $T(g, a, \perp)$ is one for all $g \in G$ and all $a \in A(g)$. The set G can thus be taken to define the action-cost function C, as well as constrain the transition probabilities T.

A (stochastic) *policy* for an MDP $\pi : S \times \mathbb{N} \to \mathcal{P}(A)$ specifies a distribution over actions for each state at each finite horizon. The cost-to-go function $J^\pi(s, k)$ gives the expected cumulative cost for k steps of execution starting at state s selecting actions according to $\pi()$ at each state encountered. For any horizon k, there is at least one (deterministic) optimal policy $\pi^*(\cdot, k)$ for which $J^{\pi^*}(s, k)$, abbreviated $J^*(s, k)$, is no greater than $J^\pi(s, k)$ at every state s, for any other policy π. The following "Q function" evaluates an action a by using a provided cost-to-go function J to estimate the value after action a is applied,

$$Q(s, a, J) = \sum_{s' \in S} T(s, a, s')[C(s, a, s') + J(s')].$$

Recursive Bellman equations use $Q()$ to describe J^* and J^π as follows:

$$J^\pi(s, k) = E\left[Q(s, \pi(s, k), J^\pi(\cdot, k-1))\right] \text{ and}$$
$$J^*(s, k) = \min_{a \in A(s)} Q(s, a, J^*(\cdot, k-1)),$$

taking the expectation over the random choice made by the possibly stochastic policy $\pi(s, k)$. In both cases, the zero step cost-to-go function is zero everywhere, so that $J^*(s, 0) = J^\pi(s, 0) = 0$ for all s. *Value iteration* computes $J^*(s, k)$ for each k in increasing order starting at zero. Note that when a policy or cost function does not depend on k, we may drop k from its argument list.

Also using $Q()$, we can select an action greedily relative to any cost function. The policy Greedy(J) selects, at any state s and horizon k, a uniformly randomly selected action from $\text{argmin}_{a \in A(s)} Q(s, a, J(\cdot, k-1))$.

While goal-based MDP problems can be directly specified as above, they may also be specified exponentially more compactly using planning languages such as PPDDL (Younes, Littman, Weissman, & Asmuth, 2005), as used in our experiments. Our technique below avoids converting the entire PPDDL problem explicitly into the above form, for resource reasons, but instead constructs a sequence of smaller problems of explicit MDP form modeling heuristic flaws.

A *dead-end state* is a state for which every policy has zero probability of reaching the goal at any horizon. We say that a policy *reaches a region of states with probability one* if following that policy to horizon k has a probability of entering the region at some point that converges to one as k goes to infinity. We say *dead-ends are unavoidable* in a problem whenever there is no policy from s_{init} that reaches the goal region with probability one. (We then say a domain has unavoidable dead-ends if any problem in that domain has unavoidable dead-ends.) We note that greedy techniques such as hill-climbing can be expected to perform poorly in domains that have dead-end states with attractive heuristic values. Application of SEH thus leaves the responsibility for detecting and avoiding dead-end states in the design of the heuristic function.

A heuristic $h : S \to \mathcal{R}$ may be provided, intended as an estimate of the cost function J for large horizons, with $h(s) = 0$ for $s \in G$, and $h(s) > 0$ otherwise. The heuristic may indicate dead-end states by returning a large positive value $V_\perp$ which we assume is selected by the experimenter to exceed the expected steps to the goal from any state that can reach the goal. In our experiments, we add trivial, incomplete dead-end detection (described in Section 5.2) to each heuristic function that we evaluate.

We note that some domains evaluated in this paper do contain unavoidable deadends, so that there may be no policy with success ratio one. The choice of the large value used for recognized dead-end states effects a trade-off between optimizing success ratio and optimizing expected cost incurred to the goal when successful.

2.2 Determinizing Stochastic Planning Problems

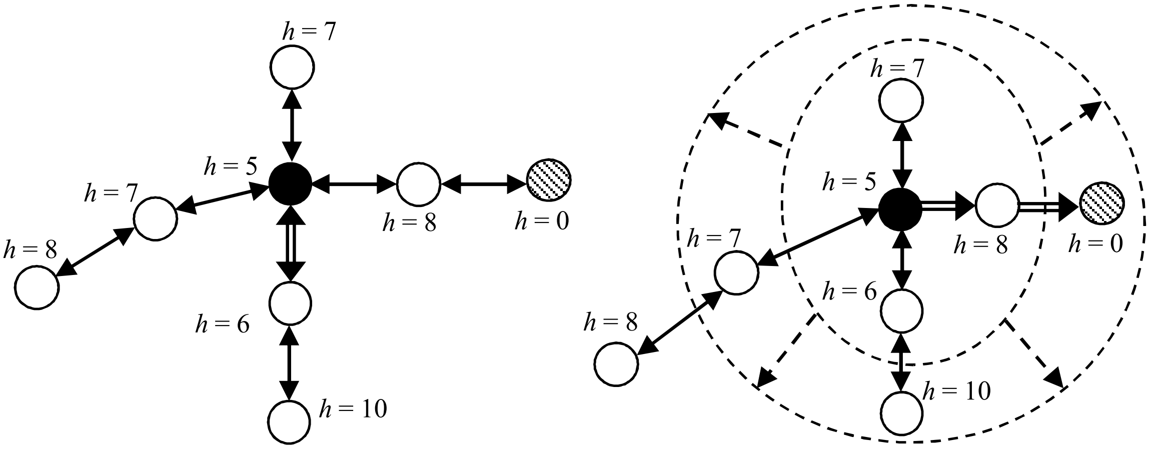
Some stochastic planners and heuristic computation techniques, including some used in our experiments, rely on computing deterministic approximations of stochastic problems. One such planner, the all-outcomes FF-Replan (Yoon et al., 2007), determinizes a stochastic planning problem and invokes the deterministic planner FF (Hoffmann & Nebel, 2001) on the determinized problem. The determinization used in FF-Replan is constructed by creating a new deterministic action for each possible outcome of a stochastic action while ignoring the probability of that outcome happening. This effectively allows the planner to control the randomness in executing actions, making this determinization a kind of relaxation of the problem. In Section 5.2, we define a domain-independent heuristic function, the "controlled-randomness FF heuristic" (CR-FF), as the deterministic FF heuristic (Hoffmann & Nebel, 2001) computed on the all-outcomes FF-Replan determinization of the probabilistic problem[3]. A variety of relaxations have previously been combined with a variety of deterministic heuristics in order to apply deterministic planning techniques to stochastic problems (Bonet & Geffner, 2005). More generally, deterministic relaxations provide a general technique for transferring techniques from deterministic planning for use in solution of stochastic problems.

3. Stochastic Enforced Hill-Climbing

Deterministic enforced hill-climbing (DEH) (Hoffmann & Nebel, 2001) searches for a successor state of strictly better heuristic value and returns a path from the current state to such a successor. This path is an action sequence that guarantees reaching the desired successor. We illustrate the behavior of DEH as compared to greedy policy using the example in Figure 1. In a stochastic environment, there may be no single better descendant that can be reached with probability one, since actions may have multiple stochastic outcomes. If we simply use breadth-first search as in DEH to find a single better descendant and ignore the other possible outcomes, we might end up selecting an action with very low probability of actually leading to any state of better heuristic value, as illustrated in Figure 2. As shown in this figure, our algorithm, stochastic enforced hill-climbing (SEH), accurately analyzes the probabilistic dynamics of the problem of improving the heuristic value.

In this section, we give details of SEH. We note that in DEH, the local breadth-first search gives a local policy in a state region surrounding the current state in a deterministic environment. The value of following this policy is the heuristic value of the improved descendant found during breadth-first search. In SEH, we implement these same ideas in a stochastic setting.

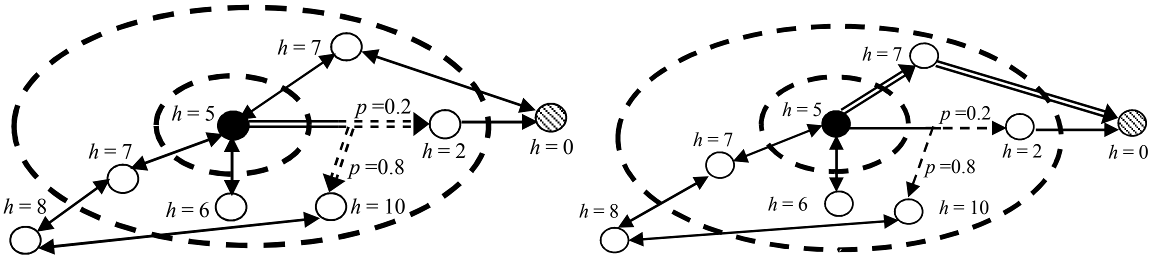
[3] The deterministic FF heuristic, described in the work of Hoffmann and Nebel (2001), from FF planner version 2.3 available at http://www.loria.fr/˜hoffmanj/ff.html, efficiently computes a greedy plan length in a problem relaxation where state facts are never deleted. The plan found in the relaxed problem is referred to as a "relaxed plan" for the problem.



(a) Behavior of greedy policy.

(b) Behavior of DEH.

Figure 1: Comparison between the behavior of DEH and greedy policy when a local optimum is encountered. The solid black circle represents the current state, and the shaded circle represents the goal state (with heuristic value zero). In (a) the greedy policy keeps selecting actions indicated by the wide arrow and cannot reach the goal state. On the other hand, DEH uses breadth-first search and finds the goal state that is two steps away from the current state, as shown in (b).



(a) Behavior of DEH in stochastic environments.

(b) Behavior of SEH in stochastic environments.

Figure 2: Comparison between the behavior of SEH and DEH in a stochastic example. We assume DEH first determinizes the problem, creating one deterministic action for each possible stochastic outcome. The solid black circle represents the current state, and the shaded circle represents the goal state (with heuristic value zero). In (a) DEH looks one step ahead and selects the action drawn with double lines, as one of the outcomes leads to a state with $h = 2$, which is better than the current state. However, this action choice has a higher probability of going to the state with $h = 10$ than the one with $h = 2$. In (b) SEH first decides there is no policy with better value than 5 when the horizon in the MDP only includes states reachable from the current state in one step. SEH then extends the horizon to two so that all states are considered. It then selects the actions indicated in the wide arrows that lead to the goal state.

```
Online Planning using a Local Planner
1. Repeat
2.     $s \leftarrow$ current state
3.     $\pi_{\text{local}} \leftarrow$ Find-Local-Policy($s$,$h$)
4.     Follow $\pi_{\text{local}}$ until $a_{\perp}$ is selected
5. Until the goal is reached
```

Table 1: Pseudo-code for an online planning framework. The policy π_{local} may be non-stationary, in which case the local planner also returns the initial horizon for execution of the policy and termination in line 4 can also happen by reaching that specified horizon.

We present SEH in two steps. First, we present a simple general framework for online planning that repeatedly calls a "local" planner that selects a policy around the current state. Second, we present a local planner based on the enforced hill-climbing idea. When the online planning framework is instantiated with this local planner, the resulting algorithm is SEH. The combination of these two steps constitute the central algorithmic contribution of this paper. Finally, we present some analytical properties of our algorithm.

3.1 A Simple Online Planning Framework

A familiar direct approach to online planning is to call the planner at the current state and have the planner select an action. That action is then executed in the environment, resulting in a new current state. This process can then be repeated.

Here, we present a simple generalization of this approach that allows the planner to select more than one action during each call, before any action is executed. The idea is that the planner makes a plan for the local context surrounding the current state, and then that plan is executed until the local context is exited. When the local context is exited, we have a new current state and the process is repeated.

More formally, we augment the action space with a new "terminate" action (called $a_{\perp}$), indicating that the planned-for local context has been exited. We then define a *local policy around a state* s to be a partial mapping from states to the augmented action space that is defined at s and at every state reachable from s under the policy[4]. An online planner can then be built by repeatedly seeking and executing a local policy around the current state using a planning subroutine. The local policy is executed until the terminate action is called (which has no effect on the state), at which point a new local policy must be sought. These ideas are reflected in the pseudo-code shown in Table 1.

We note that the notion of "local context" in our discussion here is informal — the precise notion is given by the use of the "terminate" action. A local policy is executed until it selects the "terminate" action. The **Find-Local-Policy** routine is free to use any method to decide when a state will be assigned the terminate action. Previously published *envelope* methods (Dean, Kaelbling, Kirman, & Nicholson, 1995) provide one way to address this issue, so that termination will be assigned to every state outside some "envelope" of states. However, this framework is more general than envelope methods, and allows for local policies that are not selected based upon pre-existing

[4] The local policy returned can be non-stationary and finite horizon, but must then select the terminate action at the final stage, in all reachable states.

envelopes of states (though we can always, post-planning, interpret the set of reachable states as an envelope). The general intuition is that selecting the action for the current states may involve analysis that is sufficient to select actions for many surrounding states, so our framework allows the **Find-Local-Policy** routine to return a policy specifying all such action selections.

Also, we note that this online planning framework includes recent re-planners such as FF-Replan (Yoon et al., 2007) and RFF (Teichteil-Konigsbuch et al., 2010). However, replanning because the current plan failed (e.g. because the determinization used to generate it was naive) is quite different in character from SEH, which constructs plans to improve the heuristic value, and replans each time such a plan terminates. Thus, SEH uses the heuristic function to define subgoals and plan for the original goal incrementally.

It remains to present the local planner we combine with this online planning framework to define stochastic enforced hill climbing. Our local planner analyzes the MDP problem around the current state, but with the heuristic function integrated into the problem so as to embody the subgoal of improving on the heuristic value of the current state. We describe this simple integration of the heuristic function into the problem next, and then discuss the local planner based on this integration.

3.2 Heuristic-Based Markov Decision Processes

Our method relies on finite horizon analyses of a transformed MDP problem, with increasing horizons. We transform the MDP problem with a novel *heuristic achievement transform* before analysis in order to represent the goal of finding and executing a policy that expects to improve on the initial (current) state's heuristic value.

The heuristic achievement transform is very straightforward, and applies to any goal-oriented MDP problem. First, all action costs are removed from the problem. Second, the "terminate" action $a_\perp$ is assigned the action cost $h(s)$ and transitions deterministically to the absorbing state $\perp$. When a policy is executed, the selection of the action $a_\perp$ at any state will result in replanning, as discussed in the online planning framework just presented. The actions $a_\perp$ can be thought of as "heuristic achievement" actions, allowing the immediate achievement of the value promised by the heuristic function.

Analyzing the MDP transformed by the heuristic achievement transform at a finite horizon around s_0 represents the problem of finding a policy for improving on the heuristic value of s_0 without regard to the cost of achieving such improvement in the heuristic. Allowing the heuristic achievement action $a_\perp$ to be selected at any point at any state reflects the greedy nature of this goal: the planner is not forced to look further once an improvement is found, so long as there is a policy from the initial state that expects to see improvement.

Formally, given MDP $M = (S, A, C, T, s_0)$ and non-negative heuristic function $h : S \to \mathbb{R}$, the *heuristic achievement transform of M under h, written M_h,* is given by (S, A', C', T', s_0), where A', C', and T' are as follows. Let s, s_1, and s_2 be arbitrary states from S. We define $A'(s)$ to be $A(s) \cup \{a_\perp\}$, and we take $C'(s_1, a, s_2) = 0$ and $T'(s_1, a, s_2) = T(s_1, a, s_2)$ for each $a \in A(s_1)$. Finally, we define $T'(s, a_\perp, \perp) = 1$ and $C'(s, a_\perp, \perp) = h(s)$.

The transformed MDP will have zero-cost policies at all states, and as such is not of immediate use. However, policies that are required to select $a_\perp$ as their final action (at horizon one) represent policies that are seeking to get to regions with low heuristic value, whatever the cost. An increasing-horizon search for such policies corresponds roughly to a breadth-first search for an improved heuristic value in deterministic enforced hill-climbing. Formally, we define the class of heuristic-achievement policies H as the class of policies $\pi(s,k)$ that satisfy $\pi(s,1) = a_\perp$ for all s. We define $J_h^*(s,k)$ to be the value $\min_{\pi \in H} J^\pi(s,k)$ in the heuristic transform MDP for h and π_h^* to be a policy that achieves this value. We note that, due to the zeroing of non-terminal action costs, $J_h^*(s,k)$ represents the expected heuristic value achieved at the next execution of $a_\perp$, where $a_\perp$ is required at horizon k if not before. Formally, if we define the random variable s' to be the state at which π_h^* first executes $a_\perp$ in a trajectory from s, we have $J_h^*(s,k) = E[h(s')]$.

The rough motivation for setting action costs to zero for the analysis of the heuristic-based MDP is that such actions are being considered by our method to remediate a flawed heuristic. The cumulative action cost required to reach a state of improved heuristic value is a measure of the magnitude of the flaw in the heuristic. Here, we remove this cost from the analysis in order to directly express the subgoal of “reaching a state with lower heuristic value”. Including action costs might, for example, lead to preferring cheap paths to higher heuristic values (i.e., to states worse than s_0) when expensive paths to lower heuristic values have been found. The basic motivation for enforced hill climbing is to strongly seek improved heuristic values. Instead of diluting this subgoal by adding in action costs, our methods seek the “shortest” path to a heuristic improvement by analyzing the heuristic-based MDP with an iteratively deepened finite horizon, as discussed in the next subsection. This approach is most reasonable in settings where each action has the same cost, so that the finite-horizon value iteration is a stochastic-setting analogue to uniform cost search. In settings with varying action cost, future work is needed to adapt SEH to usefully consider that cost without excessively diluting the focus on improving the heuristic.

3.2.1 HEURISTIC ACHIEVEMENT VALUE ITERATION

Following the formalism of value iteration in Section 2.1, we compute $J_h^*(s,k)$ with *heuristic achievement value iteration* as follows:

$$J_h^*(s,1) = h(s), \text{ and}$$
$$J_h^*(s,k) = \min_{a \in A'(s)} Q(s,a,J_h^*(\cdot,k-1)) \text{ for } k \geq 2.$$

A non-stationary policy achieving the cost-to-go given by $J_h^*(\cdot,k)$ can also be computed using the following definition:

$$\pi_h^*(s,1) = a_\perp, \text{ and}$$
$$\pi_h^*(s,k) = \text{argmin}_{a \in A'(s)} Q(s,a,J_h^*(\cdot,k-1)) \text{ for } k \geq 2.$$

Note that $Q()$ is computed on the heuristic-achievement transformed MDP M_h in both equations. For technical reasons that arise when zero-cost loops are present, we require that tie breaking in the argmin for $\pi_h^*(s,k)$ favors the action selected by $\pi_h^*(s,k-1)$ whenever it is one of the options. This is to prevent the selection of looping actions over shorter, more direct routes of the same value.

3.3 A Local Planner

We consider a method to be stochastic enforced hill-climbing if it uses an online planning framework, such as that presented in Table 1, together with a local policy selection method that solves the heuristic-achievement MDP (exactly, approximately, or heuristically). Here, we describe one straightforward method of local policy selection by defining **SEH-find-local-policy** using finite-horizon value iteration. This method generalizes the breadth-first search used in deterministic enforced hill-climbing, and seeks an expected heuristic improvement rather than a deterministic path to an improved heuristic value. More sophisticated and heuristic methods than finite-horizon value iteration should be considered if the implementation presented here finds the local MDP problems intractable. Our analytical results in Section 3.4 apply to any method that exactly solves the heuristic-achievement MDP, such as the method presented in Table 2; our experimental results are conducted using the implementation in Table 2 as well.

We present pseudo-code for **SEH-Find-Local-Policy** in Table 2. A heuristic function h *respects goals* if $h(s) = 0$ iff $s \in G$. The algorithm assumes a non-negative heuristic function $h : S \to \mathbb{R}$ that respects goals, as input. SEH-Find-Local-Policy(s_0,h) returns the policy π_h^* and a horizon k. The policy is only computed for states and horizons needed in order to execute π_h^* from s_0 using horizon k until the policy terminates.

Thus, in lines 5 to 11 of Table 2 , heuristic-achievement value iteration is conducted for increasing horizons around s_0, seeking a policy improving $h(s_0)$. Note that for a given horizon $k + 1$, only states reachable within k steps need to be included in the value iteration.

3.3.1 EARLY TERMINATION

The primary termination condition for repeated local policy construction is the discovery of a policy improving on the heuristic estimate of the initial state. As discussed below in Proposition 1, in domains without deadends, SEH-Find-Local-Policy will always find a policy improving on $h(s_0)$, given sufficient resources.

However, for badly flawed heuristic functions the "large enough" horizons that are analyzed in SEH-Find-Local-Policy may be unacceptably large given resource constraints. Moreover, in domains with unavoidable deadends, there may be no horizon, however large, with a policy improving on the heuristic at the initial state. For these reasons, in practice, the algorithm stops enlarging the horizon for heuristic-based MDP analysis when user-specified resource limits are exceeded.

When horizon-limited analysis of the heuristic-transform MDP construction does not yield the desired results inexpensively, biased random walk is used to seek a new initial state. As an example, consider a problem in which the provided heuristic labels all states reachable in k steps with cost-to-go estimates that are very similar, forming a very large plateau. If analysis of this large plateau exceeds the resources available, biased random walk is indicated, for lack of useful heuristic guidance.

So, once a horizon k is found for which $J_h^*(s_0, k) < h(s_0)$, the system executes π_h^* from s_0 at horizon k until the terminate action $a_\perp$ is selected. If the resource limit is exceeded without finding such a horizon, the system executes a biased random walk of length κ, with the terminating action $a_\perp$ imposed in all states s reachable by such biased random walk with $h(s) < h(s_0)$. This additional biased random walk allows our method to retain some of the beneficial properties of random exploration in domains where heuristic flaws are too large for MDP analysis. The resource consumption threshold at which random walk is triggered can be viewed as a parameter controlling the blend of random walk and MDP-based search that is used in overcoming heuristic flaws. We currently do not have a principled way of analyzing the tradeoff between resource consumption and the cost of switching to biased random walk, or determining when to do such switching. Instead, we use domain-independent resource limits as described in Section 5.1, which are determined after some experimentation.

```
SEH-Find-Local-Policy($s_0$,$h$)
//  $s_0$ is the current state.
//  $h : S \cup \{\perp\} \to \mathbb{R}$ the heuristic function, extended with $h(\perp) = 0$.
//  We assume global variable $M_h$ is the heuristic-achievement
    transform of the original problem $M$ under $h$.
//  We seek a policy in the problem $M_h$ achieving cost less than $h(s_0)$.
$k = 1$
Repeat
    $k = k + 1$
    // Compute $J_h^*(s_0, k)$ in $M_h$ using value iteration
    $J_h^*(\cdot, 1) = h(\cdot)$, $\pi_h^*(\cdot, 1) = a_\perp$, $n = 1$
    Repeat
        $n = n + 1$
        For each $s$ reachable from $s_0$ in $M_h$ using at most $k - n$ steps
            $J_h^*(s, n) = \min_{a \in A'(s)} Q(s, a, J_h^*(\cdot, n - 1))$
            $\pi_h^*(s, n) = \text{argmin}_{a \in A'(s)} Q(s, a, J_h^*(\cdot, n - 1))$
    Until $n = k$
Until $J_h^*(s_0, k) < h(s_0)$ or resource consumption exceeds user-set limits
If $J_h^*(s_0, k) < h(s_0)$
    then
        Return $\pi_h^*$ and horizon $k$
    else
        // Return a $\kappa$-step biased random walk policy $\pi$
        // Note: implementations should compute $\pi$ lazily online
        For n = 1 to $\kappa$
            For each state $s$
                If $h(s) < h(s_0)$
                    then
                        $\pi(s, n)$ selects $a_\perp$ with probability one
                    else
                        $\pi(s, n)$ selects each action $a \in A(s)$ with probability $\frac{e^{-Q(s,a,h)}}{\sum_{a_i \in A(s)} (e^{-Q(s,a_i,h)})}$
        Return $\pi$ and horizon $\kappa$
```

Table 2: Pseudo-code for the local planner used to implement stochastic enforced hill-climbing.

Horizon-limited analysis of the heuristic-transform MDP may also terminate without finding a horizon k such that $J_h^*(s_0, k) < h(s_0)$ when the entire reachable statespace has been explored, in the presence of deadends. This may happen without exceeding the available resources, and in this case we fall back to a fixed number of iterations of standard VI on the original MDP model (including action costs and without the heuristic transform) for the reachable states.

3.4 Analytical Properties of Stochastic Enforced Hill-Climbing

In deterministic settings, given sufficient resources and no dead-ends, enforced hill-climbing can guarantee finding a deterministic path to an improved heuristic value (if nothing else, a goal state will suffice). Given the finite state space, this guarantee implies a guarantee that repeated enforced hill-climbing will find the goal.

The situation is more subtle for stochastic settings. In a problem with no dead-ends, for every state s the optimal policy reaches the goal with probability one. It follows that in such problems, for any h assigning zero to every goal state, for every state s and real value $\epsilon > 0$, there is some horizon k such that $J_h^*(s, k) < \epsilon$. (Recall that J_h^* analyzes the heuristic transform MDP wherein action costs are dropped except that $h()$ must be realized at horizon one.) Because SEH-Find-Local-Policy(s,h) considers each k in turn until $J_h^*(s, k) < h(s)$ we then have:

Proposition 1. *Given non-goal state s, no dead-ends, non-negative heuristic function $h : S \to \mathbb{R}$ respecting goals, and sufficient resources, the routine SEH-Find-Local-policy(s,h) returns the policy π_h^* and a horizon k with expected return $J_h^*(s, k)$ strictly less than $h(s)$.*

However, unlike the deterministic setting, the policy found by the routine SEH-Find-Local-Policy only *expects* some improvement in the heuristic value. So particular executions of the policy from the current state may result in a degraded heuristic value.

Here, we prove that even in stochastic settings, in spite of this possibility of poor results from one iteration, SEH will reach the goal region with probability one, in the absence of dead-end states and with sufficient resources. In practice, the provision of "sufficient resources" is a serious hurdle, and must be addressed by providing a base heuristic with modest-sized flaws.

Theorem 1. *In dead-end free domains, with unbounded memory resources, SEH reaches the goal region with probability one.*

Proof. Let $x_0, x_1, x_2, \ldots, x_m, \ldots$ be random variables representing the sequence of states assigned to s in line 2 of Table 1 when we execute SEH on a planning problem, with x_0 being the initial state s_{init}. If the algorithm achieves $x_\tau \in G$ for some τ, and thus terminates, we take $x_{j+1} = x_j$ for all $j \geq \tau$. (Note that as a result $x_j \in G$ implies $x_{j+1} \in G$, where G is the goal region of states.)

We show that for arbitrary $m > 0$ the probability that x_m is in the goal region is at least $1 - \frac{h(s_{\text{init}})}{m\delta}$, for a real value $\delta > 0$ defined below. This expression goes to one as m goes to infinity, and so we can conclude that SEH reaches the goal region with probability one.

By Proposition 1, from any non-goal state s, absent dead-ends and with sufficient resources, one iteration of SEH is guaranteed to return a policy for some finite horizon k_s with value $J_h^*(s, k_s)$ improving on $h(s)$. Let $\delta_s = h(s) - J_h^*(s, k_s) > 0$ be the value of the improvement from s at horizon k_s. Because there are finitely many non-goal states, there exists $\delta = \min_{s \in S-G} \ \delta_s > 0$ such that the improvement $h(s) - J_h^*(s, k_s)$ is at least δ. Consider an arbitrary i such that $x_i \notin G$. Noting that $J_h^*(x_i, k_{x_i}) = E[h(x_{i+1})]$ due to zero action costs in M_h, it follows immediately then that $E[h(x_i) - h(x_{i+1}) | x_i \notin G] \geq \delta$, where G is the goal region of states. Using that $x_i \in G$ implies both $x_{i+1} \in G$ and $h(x_i) = h(x_{i+1}) = 0$, we write this as

$$\begin{aligned} & E[h(x_i) - h(x_{i+1})] \\ =& E[h(x_i) - h(x_{i+1}) | x_i \notin G] Q_i \\ & \quad + E[h(x_i) - h(x_{i+1}) | x_i \in G](1 - Q_i) \\ \geq & Q_i \delta, \text{ for } \delta > 0, \end{aligned} \tag{1}$$

defining Q_j to be the probability that $x_j \notin G$.

Now, we lower-bound the expected heuristic improvement $E[h(x_0) - h(x_m)]$ after m calls to **SEH-Find-Local-Policy**, for $m > 0$. We can decompose this expected improvement over m calls to **SEH-Find-Local-Policy** as the sum of the expected improvements for the individual calls. Then, lower-bounding this sum using its smallest term, we get

$$\begin{aligned} & E[h(x_0) - h(x_m)] \\ =& \sum_{i=0}^{m-1} E[h(x_i) - h(x_{i+1})] \\ \geq & \sum_{i=0}^{m-1} Q_i \delta \ \text{ (from Inequality 1)} \\ \geq & \, m Q_m \delta, \end{aligned} \tag{2}$$

as Q_m is non-increasing, since $x_{m-1} \in G$ implies $x_m \in G$.

Next, we combine this lower bound with the natural upper bound $h(s_{\text{init}})$, since h is assumed to be non-negative (so $E[h(x_0) - h(x_m)] \leq h(s_{\text{init}})$) and $x_0 = s_{\text{init}}$. Thus,

$$h(s_{\text{init}}) \geq Q_m m \delta.$$

Therefore the probability Q_m that $x_m \notin G$ is at most $\frac{h(s_{\text{init}})}{m\delta}$, converging to zero with large m and so SEH reaches the goal with probability one. □

While the theorem above assumes the absence of dead-ends, problems with dead-ends are covered by this theorem as well if the dead-ends are both avoidable and identified by the heuristic. Specifically, we may require that the heuristic function assigns ∞ to a state if and only if there is no policy to reach the goal from that state with probability one. In this case, the problem can be converted to the form required by our theorem by simply removing all states assigned ∞ from consideration (either in pre-processing or during local MDP construction).

3.5 Variants and Extensions of SEH

SEH is based on finite-horizon analysis of the MDP transformed by the heuristic-achievement transform around the current state s_0. The particular heuristic-achievement transform we describe is of course not the only option for incorporating the heuristic in a local search around s_0. While we have already considered a number of related alternatives in arriving at the choice we describe, other options can and should be considered in future research. One notable restriction in our transform is the removal of action costs, which is discussed in Section 3.2. It is important for the method to retain the actual heuristic value in the analysis so that it can trade off large, small, positive and negative changes in heuristic value according to their probabilities of arising. For this reason, we do not have the heuristic transform abstract away from the value and simply assign rewards of 1 or 0 according to whether the state improves on $h(s_0)$. Our choice to remove action costs during local expansion can lead to poor performance in domains with flawed heuristics interacting badly with high variations in action costs. This is a subject for future research on the method.

Also, the MDP models we describe in this paper are limited in some obvious ways. These limitations include that the state space is discrete and finite, the problem setting lacks discounting, and the objective is goal-oriented. We have yet to implement any extension to relax these limitations, and leave consideration of the issues that arise to future work. We note that it would appear that the method is fundamentally goal-oriented, given the goal of repeatedly reducing the heuristic value of the current state. However, it is possible to contemplate infinite-horizon discounted non-goal-oriented variants that seek policies that maintain the current heuristic estimate.

3.6 Incorporating FF Goal-Ordering Techniques in SEH

The planner FF contains heuristic elements inspired by ordering issues that arise in the blocksworld problems (Hoffmann & Nebel, 2001). These heuristic elements improve performance on the blocksworld problems significantly. To assist in a fair comparison of SEH with FF-Replan, we have implemented two of the heuristic elements, namely *goal agenda* and *added goal deletion*, in a variant of SEH that we call SEH$^+$.

The implementation of SEH$^+$ is as follows. The stochastic planning problem is first determinized using the "all outcomes determinization" described in Section 2.2. The goal-agenda technique of FF is then invoked on the determinized problem to extract a sequence of temporary goals $G_1, \ldots, G_m$, where each G_i is a set of goal facts and G_m is the original problem goal. SEH with a stochastic version of added goal deletion, described next in this subsection, is then invoked repeatedly to compute a sequence of states $s_0, \ldots, s_m$, where s_0 is the initial state and for $i > 0$ each s_i is defined as the state reached by invoking SEH in state s_{i-1} with goal G_i (thus satisfying G_i).

Added goal deletion is the idea of pruning the state search space by avoiding repetitive addition and deletion of the same goal fact along searched paths. In FF, for a search state s, if a goal fact is achieved by the action arriving at s, but is deleted by an action in the relaxed plan found from s, then s is not expanded further (Hoffmann & Nebel, 2001).

For our stochastic adaptation of added goal deletion, we define the set of facts added by any state transition (s, a, s') to be those facts true in s' but not in s and represent it as the set difference $s' - s$. Then, the set of added goal facts for the transition are those added facts which are also true in the current temporary goal G_i, i.e., $(s' - s) \cap G_i$. We prune any state transition (s, a, s') whenever the relaxed plan computed from s' to the current temporary goal G_i contains an action that deletes any of the added goal facts. A transition (s, a, s') is "pruned" by modifying the Bellman update at state s so that s' contributes the dead-end state value ($V_\perp$) to the Q-value for a at s, weighted by the transition probability (instead of contributing the cost-to-go at s'). More formally, we define a modified Q-function when using added goal deletion, $Q_{agd}(s, a, J)$ as follows:

$$I(s') = \begin{cases} 1, & \text{if } f \in (s' - s) \cap G_i \text{ is deleted by an action in relaxed\_plan}(s', G_i)^5 \\ 0, & \text{otherwise} \end{cases}$$

$$Q_{agd}(s, a, J) = \sum_{s'} T(s, a, s')[I(s')V_\perp + (1 - I(s'))J(s')]$$

$Q_{agd}()$ then replaces $Q()$ in the definition of the cost-to-go function $J^*_h()$ in Section 3.2. Also, "reachability" in line 8 of Table 2 does not use pruned transitions.

In some problems, subsequent deletion of newly added goals is unavoidable for any valid plan. Added goal deletion prunes all routes leading to the goal region for such problems even though no actual deadend is present. Hence, this is an incomplete technique as discussed in the work of Hoffmann and Nebel (2001). FF falls back to best-first search if DEH is not able to find a valid plan due to pruning. Similarly, when unable to find an improved policy, SEH falls back to either value iteration or biased random walk as described in Section 3.3.

Preliminary exploration of incorporating stochastic variants of FF's helpful action pruning (Hoffmann & Nebel, 2001) into SEH did not improve performance, much like the effect of added goal deletion on all domains except the blocksworld[6]. As a result, we do not report on helpful-action-pruning methods here.

4. Related Work

In this section we discuss planning techniques that are close to our work in one or more dimensions.

4.1 Fast-Foward (FF) Planner and Deterministic Enforced Hill-Climbing

For an introduction to deterministic enforced hill-climbing (DEH) and its relation to our technique, please see Section 3. Here, we additionally note that there are several lines of work that directly extend the FF planner to allow planning with numeric state-variables (Hoffmann, 2003) and planning with uncertainty (Hoffmann & Brafman, 2006, 2005; Domshlak & Hoffmann, 2007). Although these techniques involve significant changes to the computation of the relaxed-plan heuristic and the possible addition of the use of belief states to handle uncertainty, enforced hill-climbing is still the primary search technique used in these lines of work. We note that although in the work of Domshlak and Hoffmann actions with probabilistic outcome are handled, the planner (Probabilistic-FF) is designed for probabilistic planning with no observability, whereas our planner is designed for probabilistic planning with full observability.

4.2 Envelope-Based Planning Techniques

Stochastic enforced hill-climbing dynamically constructs local MDPs to find a local policy leading to heuristically better state regions. The concept of forming local MDPs, or "envelopes", and using

[5] relaxed_plan(s',G_i) computes the relaxed plan between states s' and G_i as defined in the work of Hoffmann and Nebel (2001) using the all-outcomes problem determinization defined in Section 2.2.

[6] We explored ideas based on defining the helpfulness of an action to be the expectation of the helpfulness of its deterministic outcomes.

them to facilitate probabilistic planning has been used in previous research such as the work of Bonet and Geffner (2006), and Dean et al. (1995), which we briefly review here.

The envelope-based methods in the work of Dean et al. (1995) and Gardiol and Kaelbling (2003) start with a partial policy in a restricted area of the problem (the "envelope"), then iteratively improves the solution quality by extending the envelope and recomputing the partial policy. The typical assumption when implementing this method is that the planner has an initial trajectory from the starting state to the goal, generated by some stochastic planner, to use as the initial envelope.

Another line of work, including RTDP (Barto, Bradtke, & Singh, 1995), LAO* (Hansen & Zilberstein, 2001), and LDFS (Bonet & Geffner, 2006), starts with an envelope containing only the initial state, and then iteratively expands the envelope by expanding states. States are expanded according to state values and dynamic programming methods are used to backup state values from newly added states, until some convergence criterion is reached. Stochastic enforced hill-climbing can be viewed as repeatedly deploying the envelope method with the goal, each time, of improving on the heuristic estimate of distance-to-go. For a good h function, most invocations result in trivial one-step envelopes. However, when local optima or plateaus are encountered, the envelope may need to grow to locate a stochastically reachable set of exits.

All of the above referenced previous search methods have constructed envelopes seeking a high quality policy to the goal rather than our far more limited and relatively inexpensive goal of basin escape. Our results derive from online greedy exploitation of the heuristic rather than the more expensive offline computation of converged values proving overall (near) optimality. LDFS, for example, will compute/check values for at least all states reachable under the optimal policy (even if given J^* as input) and possibly vastly many others as well during the computation.

Some of these previous methods are able to exploit properties (such as admissibility) of the heuristic function to guarantee avoiding state expansions in some regions of the state space. Clearly, SEH exploits the heuristic function in a way that can avoid expanding regions of the statespace. However, we have not at this point conducted any theoretical analysis of what regions can be guaranteed unexpanded for particular kinds of heuristic, and such analyses may be quite difficult.

4.3 Policy Rollout

The technique of "policy rollout" (Tesauro & Galperin, 1996; Bertsekas & Tsitsiklis, 1996) uses a provided base policy π to make online decisions. The technique follows the policy $Greedy(\widetilde{V^\pi})$, where $\widetilde{V^\pi}$ is computed online by sampling simulations of the policy π.

The computation of the optimal heuristic-transform policy π_h^* in SEH has similarities to policy rollout: in each case, online decisions are made by local probabilistic analysis that leverages provided information to manage longer-range aspects of the local choice. For SEH, a heuristic function is provided while, for policy rollout, a base policy is provided. In this view, policy rollout does local analysis under the assumption that non-local execution will use the base policy π, whereas SEH does local analysis under the assumption that non-local execution will achieve the base heuristic cost estimate h.

In fact, for our goal-oriented setting, when the provided heuristic function h is stochastic (a simple generalization of what we describe in this paper), and equal to a sampled-simulation evaluation of V^π for some policy π, then SEH executes the same policy as policy rollout, assuming uniform action costs and sufficient sampling to correctly order the action choices. This claim follows because when $h = V^\pi$ there is always some action to yield an expected improvement in h in one step, in our goal-oriented setting. The need for uniform action costs in this claim may be relaxed if a variant of SEH is developed that retains action costs in the heuristic transform.

In policy rollout, only horizon-one greedy use of the sampled heuristic is needed, but the main substance of SEH is to enable the repair and use of heuristic functions with flaws that cannot be repaired at horizon one. Thus the central differences between the techniques are reflected in the ability of SEH to leverage arbitrary heuristic functions and repair flaws in those functions at larger horizons.

Policy rollout provides an elegant guarantee that the online policy selected improves on the base policy, given sufficient sampling. This result follows intuitively because the computed policy is the policy-iteration improvement of the base policy. Unfortunately, no similar guarantee is known to apply for SEH for an arbitrary heuristic function. However, policy rollout cannot be used to improve an arbitrary heuristic function either.

4.4 Local Search in Optimization

Stochastic enforced hill-climbing can be regarded as one of many local-search techniques designed to improve on greedy one-step lookahead, the most naive form of local search optimization. Here we briefly discuss connections to the method of simulated annealing, one of a large family of related local search techniques. For more detail, please see the work of Aarts and Lenstra (1997).

Simulated annealing (Kirkpatrick et al., 1983; Cerny, 1985) allows the selection of actions with inferior expected outcome with a probability that is monotone in the action q-value. The probability that an inferior action will be selected often starts high and decreases over time according to a “cooling” schedule. The ability to select inferior actions leads to a non-zero probability of escaping local optima. However, this method does not systematically search for a policy that does so. In contrast, stochastic enforced hill-climbing analyzes a heuristic-based MDP at increasing horizons to systematically search for policies that give improved expected value (hence leaving the local extrema). In our substantial preliminary experiments, we could not find successful parameter settings to control simulated annealing for effective application to online action selection in goal-directed stochastic planning. To our knowledge, simulated annealing has not otherwise been tested on direct forward-search action selection in planning, although variants have been applied with some success to other planning-as-search settings (Selman, Kautz, & Cohen, 1993; Kautz & Selman, 1992; Gerevini & Serina, 2003) such as planning via Boolean satisfiability search.

5. Setup for Empirical Evaluation

Here, we describe the parameters used in evaluating our method, the heuristics we will test the method on, the problem categories in which the tests will be conducted, the random variables for aggregated evaluation, and issues arising in interpreting the results and their statistical significance. We run our experiments on Intel Xeon 2.8GHz machines with 533 MHz bus speed and 512KB cache.

5.1 Implementation Details

If at any horizon increase no new states are reachable, our implementation of SEH simply switches to an explicit statespace method to solve the MDP formed by the reachable states. More specifically, if the increase in k at line 3 in Table 2 does not lead to new reachable states in line 8, we trigger value iteration on the states reachable from s_0.

Throughout our experiments, the thresholds used to terminate local planning in line 12 of Table 2 are set at $1.5 * 10^5$ states and one minute. We set the biased random walk length κ to ten. This work makes the assumption that the heuristic functions used assign large values to easily recognized dead-ends, as hill-climbing works very poorly in the presence of dead-end attractor states. We enforce this requirement by doing very simple dead-end detection on the front-end of each heuristic function (described next in Section 5.2 for each heuristic) and assigning the value $1.0 * 10^5$ to recognized dead-end states.

We denote this implementation running on heuristic h with **SEH(**h**)**.

5.2 Heuristics Evaluated

We describe two different types of heuristic functions used in our evaluation and the associated dead-end detection mechanisms.

5.2.1 THE CONTROLLED-RANDOMNESS FF HEURISTIC

For use in our evaluations, we define a domain-independent heuristic function, the "controlled-randomness FF heuristic" (CR-FF). We define CR-FF on a state s to be the FF distance-to-goal estimate (Hoffmann & Nebel, 2001) of s computed on the all-outcomes determinization as described in Section 2.2. We denote the resulting heuristic function F. While computing the CR-FF heuristic, we use the reachability analysis built into the FF planner for the detection of deadends.

5.2.2 LEARNED HEURISTICS

We also test stochastic enforced hill-climbing on automatically generated heuristic functions from the work of Wu and Givan (2010), which on their own perform at the state-of-the-art when used to construct a greedy policy. We shift these heuristic functions to fit the non-negative range requirement of h discussed previously. These learned heuristic functions are currently available for only seven of our test categories, and so are only tested in those categories.

We note that these heuristics were learned for a discounted setting without action costs and so are not a direct fit to the "distance-to-go" formalization adopted here. We are still able to get significant improvements from applying our technique. We denote these heuristics L. Only states with no valid action choice available are labeled as deadends when applying SEH to the learned heuristics.

5.3 Goals of the Evaluation

Our primary empirical goal is to show that stochastic enforced hill-climbing generally improves significantly upon greedy following of the same heuristic (using the policy Greedy(h) as described in the technical background above). We will show that this is true for both of the heuristics defined in Section 5.2. We show empirically the applicability and limitation of SEH discussed in Section 1.1, in different types of problems including probabilistically interesting ones (Little & Thiebaux, 2007).

A secondary goal of our evaluation is to show that for some base heuristics the resulting performance is strong in comparison to the deterministic replanners FF-Replan (Yoon et al., 2007) and RFF (Teichteil-Konigsbuch et al., 2010). While both FF-Replan and RFF use the Fast-Forward (FF) base planner, RFF uses a most-probable-outcome determinization in contrast to the all-outcomes determinization used by FF-Replan. The primary other difference between RFF and FF-Replan is that before executing the plan, RFF grows policy trees to minimize the probability of having to replan, while FF-Replan does not.

5.4 Adapting IPPC Domains for Our Experiments

We conduct our empirical evaluation using all problems from the first three international probabilistic planning competitions (IPPCs) as well as all twelve ”probabilistically interesting” problems from the work of Little and Thiebaux (2007). We omit some particular problems or domains from particular comparisons for any of several practical reasons, detailed in an online appendix.

Because enforced hill-climbing is by nature a goal-oriented technique and SEH is formulated for the goal-oriented setting, we ignore the reward structure (including action and goal rewards) in any of the evaluated problems and assume an uniform action cost of one for those problems, transforming any reward-oriented problem description into a goal-oriented one.

We provide detailed per-problem results in an online appendix for each planner evaluated in this work. However, in support of our main conclusions, we limit our presentation here to aggregations comparing pairs of planners over sets of related problems. For this purpose, we define seventeen problem categories and aggregate within each problem category. While some categories are single domains, more generally, multiple closely related domains may be aggregated within a single category. For example, the blocksworld category aggregates all blocksworld problems from the three competitions, even though the action definitions are not exactly the same in every such problem. For some paired comparisons, we have aggregated the results of all problems labeled as or constructed to be “probabilistically interesting” by the IPPC3 organizers or by the work of Little and Thiebaux (2007) into a combined category PI PROBLEMS.

In Table 3, we list all evaluated categories (including the combined category PI PROBLEMS), as well as the planning competitions or literature the problems in each category are from. The evaluated problems in each category are identified in an online appendix.

The reward-oriented SYSADMIN domain from IPPC3 was a stochastic longest-path problem where best performance required avoiding the goal so as to continue accumulating reward as long as possible (Bryce & Buffet, 2008). (Note that contrary to the organizers’ report, the domain’s goal condition is “all servers up” rather than “all servers down”.) Our goal-oriented adaptation removes the longest-path aspect of the domain, converting it to a domain where the goal is to get all the servers up.

The BLOCKSWORLD problems from IPPC2 contain flawed definitions that may lead to a block stacking on top of itself. Nevertheless, the goal of these problems is well defined and is achievable using valid actions, hence the problems are included in the BLOCKSWORLD category.

We have discovered that the five rectangle-tireworld problems (p11 to p15 from IPPC3 2-TIREWORLD) have an apparent bug—no requirement to remain “alive” is included in the goal condition. The domain design provides a powerful teleport action to non-alive agents intended only to increase branching factor (Buffet, 2011). However, lacking a requirement to be alive in the goal, this domain is easily solved by deliberately becoming non-alive and then teleporting to the goal. We have modified these problems to require the predicate “alive”’ in the goal region. We have merged these modified rectangle-tireworld problems with triangle-tireworld problems from both IPPC3 and

Category	Problem Source(s)
Blocksworld	IPPC1, IPPC2, IPPC3
Boxworld	IPPC1, IPPC3
Busfare	Little and Thiebaux (2007)
Drive	IPPC2
Elevator	IPPC2
Exploding Blocksworld	IPPC1, IPPC2, IPPC3
Fileworld	IPPC1
Pitchcatch	IPPC2
Random	IPPC2
River	Little and Thiebaux (2007)
Schedule	IPPC2, IPPC3
Search and Rescue	IPPC3
Sysadmin	IPPC3
Systematic-tire	Triangle-tireworld (IPPC3 2-Tireworld P1 to P10, Little and Thiebaux (2007)), Rectangle-tireworld (IPPC3 2-Tireworld P11 to P15) with bug fixed
Tireworld	IPPC1, IPPC2
Towers of Hanoi	IPPC1
Zenotravel	IPPC1, IPPC2
PI problems	Busfare, Drive, Exploding Blocksworld Pitchcatch, River, Schedule, Systematic-tire, Tireworld

Table 3: List of categories and the planning competitions or literature from which the problems in each category are taken.

the work of Little and Thiebaux (2007) into a category SYSTEMATIC-TIRE, as these problems have been systematically constructed to emphasize PI features.

5.5 Aggregating Performance Measurements

For our experiments, we have designed repeatable aggregate measurements that we can then sample many times in order to evaluate statistical significance. We now define the random variables representing these aggregate measurements and describe our sampling process, as well as our method for evaluating statistical significance.

5.5.1 DEFINING AND SAMPLING AGGREGATE-MEASUREMENT RANDOM VARIABLES

For each pair of compared planners, we define four random variables representing aggregate performance comparisons over the problems in each category. Each random variable is based upon a sampling process that runs each planner five times on all problems in a category, and aggregates the per-problem result by computing the mean. We use five-trial runs to reduce the incidence of low-success planners failing to generate a plan length comparison. Each mean value from a five-trial run is a sample value of the respective random variable.

First, the per-problem success ratio (SR) is the fraction of the five runs that succeed for each problem. The success ratio random variable for each category and planner is then the mean SR across all problems in the category.

Second, the per-problem successful plan length (SLen) is the mean plan length of all successful runs among the five runs. In order to compare two planners on plan length, we then define the per-problem ratio of jointly successful plan lengths (JSLEN-RATIO) for the two compared planners as follows. If both planners have positive SR among the five trials on the problem, JSLEN-RATIO is the ratio of the SLen values for the two planners; otherwise, JSLEN-RATIO is undefined for that problem. We use ratio of lengths to emphasize small plan length differences more in short solutions than in long solutions, and to decrease sensitivity to the granularity of the action definitions.

The mean JSLEN-RATIO random variable for each category and pair of planners is then the geometric mean of the JSLEN-RATIO across all problems in the category for which JSLEN-RATIO is well defined. In this manner we ensure that the two planners are compared on exactly the same set of problems. Note then that, unlike SR, JSLEN-RATIO depends on the pair of compared planners, rather than being a measurement on any single planner; it is the ratio of successful plan length on the *jointly solved* problems for the two planners.

Similarly, the per-problem ratio of jointly successful runtimes (JSTIME-RATIO) is defined in the same manner used for comparing plan lengths. The mean JSTIME-RATIO is again computed as the geometric mean of well-defined per-problem JSTIME-RATIO values.

Because JSLEN-RATIO and JSTIME-RATIO are ratios of two measurements, we use the geometric mean to aggregate per-problem results to generate a sample value, whereas we use arithmetic mean for the SR variables. Note that geometric mean has the desired property that when the planners are tied overall (so that the geometric mean is one), the mean is insensitive to which planner is given the denominator of the ratio.

Thus, to draw a single sample of all four aggregate random variables (SR for each planner, JSLEN-RATIO, and JSTIME-RATIO) in comparing two planners, we run the two planners on each problem five times, computing per-problem values for the four variables, and then take the (arithmetic or geometric) means of the per-problem variables to get one sample of each aggregate variable. This process is used repeatedly to draw as many samples as needed to get significant results.

We use a plan-length cutoff of 2000 for each attempt. Each attempt is given a time limit of 30 minutes.

5.5.2 SIGNIFICANCE OF PERFORMANCE DIFFERENCES BETWEEN PLANNERS

Our general goal is to order pairs of planners in overall performance on each category of problem. To do this, we must trade off success rate and plan length. We take the position that a significant advantage in success rate is our primary goal, with plan length used only to determine preference among planners when success rate differences are not found to be significant.

We determine significance for each of the three performance measurements (SR, JSLEN-RATIO, and JSTIME-RATIO) using t-tests, ascribing significance to the results when the p-value is less than 0.05. The exact hypothesis tested and form of t-test used depends on the performance measurement, as follows:

1. SR — We use a paired one-sided t-test on the hypothesis that the difference in true means is larger than 0.02.

2. JSLEN-RATIO — We use a one-sample one-sided t-test on the hypothesis that the true geometric mean of JSLEN-RATIO exceeds 1.05 (log of the true mean of JSLEN-RATIO exceeds $\log(1.05)$).

3. JSTIME-RATIO — We use a one-sample one-sided t-test on the hypothesis that the true geometric mean of JSTIME-RATIO exceeds 1.05 (log of the true mean of JSTIME-RATIO exceeds $\log(1.05)$).

We stop sampling the performance variables when we have achieved one of the following criteria, representing “an SR winner is determined” or “SR appears tied”:

1. Thirty samples have been drawn and the p-value for SR difference is below 0.05 or above 0.5.

2. Sixty samples have been drawn and the p-value for SR difference is below 0.05 or above 0.1.

3. One hundred and fifty samples have been drawn.

In all the experiments we present next, this stopping rule leads to only 30 samples being drawn unless otherwise mentioned. Upon stopping, we conclude a ranking between the planners (naming a “winner”) if either the SR difference or the JSLEN-RATIO has p-value below 0.05, with significant SR differences being used first to determine the winner. If neither measure is significant upon stopping, we deem the experiment inconclusive.

Combining categories For some of our evaluations, we aggregate results across multiple categories of problem, e.g., the combined category PI PROBLEMS. In such cases, we have effectively defined one larger category, and all our techniques for defining performance measurements and determining statistical significance are the same as in Section 5.5. However, we do not actually re-run planners for such combined-category measurements. Instead, we re-use the planner runs used for the single-category experiments. Rather than use the stopping rule just described, we compute the maximum number of runs available in all the combined categories and use that many samples of the combined-category performance measurements. To avoid double counting problem results, we treat combined categories separately when analyzing the results and counting wins and losses.

6. Empirical Results

We present the performance evaluation of stochastic enforced hill-climbing (SEH) in this section. The experiments underlying the results presented here involve 169,850 planner runs in 17 categories.

6.1 Summary of Comparison

The results in Table 4 show that, for the CR-FF heuristic, SEH with the goal-ordering and added-goal-deletion enhancements (SEH$^+$(F)) improves significantly over the baseline SEH technique (SEH(F)) in the category BLOCKSWORLD, but does not show significant changes in the aggregated performance for non-blocksworld problems[7]. For the remainder of the experiments involving CR-FF, we evaluate only SEH$^+$(F), noting that both of our comparison planners (FF-Replan and RFF) benefit from the goal-ordering and added-goal-deletion enhancements of their base planner, FF-plan.

The results we present next for SEH$^+$(F) show:

- SEH$^+$(F) significantly outperforms Greedy(F) in 13 categories, but is outperformed by Greedy(F) in SCHEDULE. There were three categories where the comparison was inconclusive (BUSFARE, RIVER and TIREWORLD). See Table 5 for details.

- FF-Replan was inapplicable in two categories (IPPC3 SEARCH-AND-RESCUE and IPPC3 SYSADMIN). SEH$^+$(F) significantly outperforms FF-Replan in 10 categories, but is outperformed by FF-Replan in three categories (EXPLODING BLOCKSWORLD, PITCHCATCH, and ZENOTRAVEL). There were two categories where the comparison was inconclusive (FILEWORLD and RIVER). SEH$^+$(F) also significantly outperforms FF-Replan on the combined category PI PROBLEMS, although the winner varied between the aggregated categories. See Table 6 for details.

- RFF-BG was inapplicable in two categories (BUSFARE and IPPC1 FILEWORLD). SEH$^+$(F) significantly outperforms RFF-BG in 12 categories, but is outperformed by RFF-BG in two categories (EXPLODING BLOCKSWORLD and SYSTEMATIC-TIRE). There was one category where the comparison was inconclusive (SYSADMIN). SEH$^+$(F) also significantly outperforms RFF-BG on the combined category PI PROBLEMS, although the winner varied between the aggregated categories. See Table 7 for details.

The “learned heuristic” from the work of Wu and Givan (2010) has been computed only in a subset of the domains, hence only seven categories are applicable for the evaluation using the learned heuristic (see an online appendix for details). The results we present next for SEH with the learned heuristic, SEH(L), show:

- SEH(L) significantly outperforms Greedy(L) in six categories. There was one category (TIREWORLD) where the comparison was inconclusive. See Table 8 for details.

[7] We show p-values rounded to two decimal places. For example, we show p=0.00 when the value of p rounded to two decimal places is 0.

Category	SR of SEH$^+(F)$	SR of SEH(F)	JSLEN-RATIO (SEH/ SEH$^+$)	JSTIME-RATIO (SEH/ SEH$^+$)	SR Difference Significant? (p-value)	JSLEN-RATIO Significant? (p-value)	Winner
BLOCKSWORLD	0.93	0.72	1.58	2.85	YES (p=0.00)	YES (p=0.00)	SEH$^+(F)$
NON-BLOCKSWORLD	0.69	0.69	1.01	0.97	NO (p=1.00)	NO (p=1.00)	Inconclusive

Table 4: Aggregated comparison of SEH$^+(F)$ against SEH(F).

- SEH(L) significantly outperforms FF-Replan in five categories, but is outperformed by FF-Replan in two categories (EXPLODING BLOCKSWORLD and ZENOTRAVEL). See Table 9 for details.

6.2 Discussion

We now discuss the results for comparisons between pairs of planners, including SEH versus greedy heuristic-following, SEH versus FF-Replan, and SEH versus RFF-BG.

6.2.1 SEH/SEH$^+$ VERSUS GREEDY

Our primary evaluation goal was to show that stochastic enforced hill-climbing generally improves significantly upon greedy following of the same heuristic (using the policy Greedy(h) as described in the technical background above). This was demonstrated by evaluating SEH with two different heuristics in Tables 5 and 8, where SEH(h) significantly outperforms Greedy(h) in nineteen out of twenty-four heuristic/category pairs, only losing in SCHEDULE for SEH$^+(F)$ against Greedy(F). We now discuss the only category where Greedy outperforms SEH techniques significantly.

In SCHEDULE, there are multiple classes of network packets with different arrival rates. Packets have deadlines, and if a packet is not served before its deadline, the agent encounters a class-dependent risk of “death” as well as a delay while the packet is cleaned up. To reach the goal of serving a packet from every class, the agent must minimize the dropping-related risk of dying while waiting for an arrival in each low-arrival-rate class. The all-outcomes determinization underlying the CR-FF heuristic gives a deterministic domain definition where dying is optional (never chosen) and unlikely packet arrivals happen by choice, leading to a very optimistic heuristic value. When using a very optimistic heuristic value, the basic local goal of SEH, which is to improve on the current state heuristic, leads to building very large local MDPs for analysis. In the presence of dead-ends (“death”, as above), even arbitrarily large local MDPs may not be able to achieve a local improvement, and so in SCHEDULE, SEH$^+$ will typically hit the resource limit for MDP size at every action step.

In contrast, greedy local decision making is well suited to packet scheduling. Many well known packet scheduling policies (e.g. “earliest deadline first” or “static priority” in the work of Liu & Layland, 1973) make greedy local decisions and are practically quite effective. In our experiments, the Greedy policy applied to CR-FF benefits from locally seeking to avoid the incidental delays of dropped-packet cleanup: even though the heuristic sees no risk-of-dying cost to dropping, it still recognizes the delay of cleaning up lost dropped packets. Thus, Greedy(F) is a class-insensitive

Category	SR of SEH$^+(F)$	SR of Greedy(F)	JSLEN-RATIO (Greedy/ SEH$^+$)	JSTIME-RATIO (Greedy/ SEH$^+$)	SR Difference Significant? (p-value)	JSLEN-RATIO Significant? (p-value)	Winner
BLOCKSWORLD	0.93	0.35	1.40	0.63	YES (p=0.00)	YES (p=0.00)	SEH$^+(F)$
BOXWORLD	0.99	0.05	1.18	1.12	YES (p=0.00)	YES (p=0.00)	SEH$^+(F)$
BUSFARE	1.00	0.99	0.85	0.86	NO (p=0.97)	NO (p=0.21)	Inconclusive
DRIVE	0.69	0.35	1.60	1.41	YES (p=0.00)	YES (p=0.00)	SEH$^+(F)$
ELEVATOR	1.00	0.40	1.82	1.81	YES (p=0.00)	YES (p=0.00)	SEH$^+(F)$
EXPLODING BLOCKSWORLD	0.44	0.18	1.01	0.63	YES (p=0.00)	NO (p=0.93)	SEH$^+(F)$
FILEWORLD	1.00	0.21	1.03	0.24	YES (p=0.00)	NO (p=1.00)	SEH$^+(F)$
PITCHCATCH	0.45	0.00	–	–	YES (p=0.00)	–	SEH$^+(F)$
RANDOM	0.99	0.94	1.76	0.59	YES (p=0.00)	YES (p=0.00)	SEH$^+(F)$
RIVER	0.66	0.67	0.97	0.98	NO (p=0.60)	NO (p=0.75)	Inconclusive
SCHEDULE	0.54	0.60	1.18	0.32	YES (p=0.00)	YES (p=0.01)	Greedy(F)
SEARCH AND RESCUE	1.00	1.00	1.23	1.08	NO (p=1.00)	YES (p=0.00)	SEH$^+(F)$
SYSADMIN	0.27	0.27	1.21	1.23	NO (p=1.00)	YES (p=0.00)	SEH$^+(F)$
SYSTEMATIC -TIRE	0.29	0.21	1.03	0.72	YES (p=0.00)	NO (p=0.86)	SEH$^+(F)$
TIREWORLD	0.91	0.90	0.96	0.79	NO (p=0.93)	NO (p=0.74)	Inconclusive
TOWERS OF HANOI	0.53	0.00	–	–	YES (p=0.00)	–	SEH$^+(F)$
ZENOTRAVEL	0.90	0.20	1.31	0.74	YES (p=0.00)	YES (p=0.00)	SEH$^+(F)$

Table 5: Aggregated comparison of SEH$^+(F)$ against Greedy(F). The RIVER domain evaluation required extending sampling to 60 samples as per the experimental protocol described in Section 5.5.2. The values and p-values of JSLEN-RATIO and JSTIME-RATIO in PITCHCATCH and TOWERS OF HANOI are not available due to the zero success ratio of Greedy(F) in these categories.

Category	SR of SEH$^{+}(F)$	SR of FF-Replan	JSLEN-RATIO (FFR/ SEH$^{+}(F)$)	JSTIME-RATIO (FFR/ SEH$^{+}(F)$)	SR Difference Significant? (p-value)	JSLEN-RATIO Significant? (p-value)	Winner
BLOCKSWORLD	0.93	0.87	1.33	1.17	YES (p=0.00)	YES (p=0.00)	SEH$^{+}(F)$
BOXWORLD	0.99	0.88	3.93	1.57	YES (p=0.00)	YES (p=0.00)	SEH$^{+}(F)$
BUSFARE	1.00	0.01	0.00	0.00	YES (p=0.00)	–	SEH$^{+}(F)$
DRIVE	0.69	0.54	1.26	2.42	YES (p=0.00)	YES (p=0.00)	SEH$^{+}(F)$
ELEVATOR	1.00	0.93	0.95	0.93	YES (p=0.00)	NO (p=0.36)	SEH$^{+}(F)$
EXPLODING BLOCKSWORLD	0.44	0.44	0.85	0.56	NO (p=0.96)	YES (p=0.00)	FF-Replan
FILEWORLD	1.00	1.00	0.97	0.57	NO (p=1.00)	NO (p=1.00)	Inconclusive
PITCHCATCH	0.45	0.51	2.78	0.21	YES (p=0.00)	YES (p=0.00)	FF-Replan
RANDOM	0.99	0.96	1.37	0.19	YES (p=0.00)	YES (p=0.00)	SEH$^{+}(F)$
RIVER	0.66	0.65	0.94	0.93	NO (p=0.60)	NO (p=0.33)	Inconclusive
SCHEDULE	0.54	0.48	1.04	0.10	YES (p=0.00)	NO (p=0.59)	SEH$^{+}(F)$
SYSTEMATIC -TIRE	0.29	0.07	0.36	0.38	YES (p=0.00)	YES (p=0.00)	SEH$^{+}(F)$
TIREWORLD	0.91	0.69	0.69	0.57	YES (p=0.00)	YES (p=0.00)	SEH$^{+}(F)$
TOWERS OF HANOI	0.59	0.50	0.64	0.06	YES (p=0.00)	YES (p=0.00)	SEH$^{+}(F)$
ZENOTRAVEL	0.90	1.00	0.70	0.10	YES (p=0.00)	YES (p=0.00)	FF-Replan
PI PROBLEMS	0.55	0.45	1.02	0.54	YES (p=0.00)	NO (p=1.00)	SEH$^{+}(F)$

Table 6: Aggregated comparison of SEH$^{+}(F)$ against FF-Replan (FFR). The RANDOM and RIVER domains required extending sampling to 60 samples and the TOWERS OF HANOI domain required extending sampling to 150 samples as per the experimental protocol described in Section 5.5.2. The p-value of JSLEN-RATIO in BUSFARE is not available because the extremely low success rate of FFR leads to only one sample of JSLEN being gathered in 30 attempts, yielding no estimated variance.

Category	SR of SEH$^+(F)$	SR of RFF-BG	JSLEN-RATIO (RFF-BG/ SEH$^+(F)$)	JSTIME-RATIO (RFF-BG/ SEH$^+(F)$)	SR Difference Significant? (p-value)	JSLEN-RATIO Significant? (p-value)	Winner
Blocksworld	0.93	0.77	0.79	0.22	yes (p=0.00)	yes (p=0.00)	SEH$^+(F)$
Boxworld	0.99	0.89	1.03	3.70	yes (p=0.00)	no (p=1.00)	SEH$^+(F)$
Drive	0.69	0.61	1.07	1.24	yes (p=0.00)	no (p=0.08)	SEH$^+(F)$
Elevator	1.00	1.00	1.27	0.15	no (p=1.00)	yes (p=0.00)	SEH$^+(F)$
Exploding Blocksworld	0.44	0.43	0.84	0.56	no (p=0.92)	yes (p=0.00)	RFF-BG
Pitchcatch	0.45	0.00	–	–	yes (p=0.00)	–	SEH$^+(F)$
Random	0.99	0.74	1.26	0.56	yes (p=0.00)	yes (p=0.00)	SEH$^+(F)$
River	0.66	0.51	0.77	0.21	yes (p=0.00)	yes (p=0.00)	SEH$^+(F)$
Schedule	0.54	0.43	1.06	0.08	yes (p=0.00)	no (p=0.40)	SEH$^+(F)$
Search and Rescue	1.00	0.01	2.99	0.86	yes (p=0.00)	yes (p=0.00)	SEH$^+(F)$
Sysadmin	0.27	0.27	1.10	9.31	no (p=1.00)	no (p=0.05)	Inconclusive
Systematic -tire	0.29	0.81	1.22	4.49	yes (p=0.00)	yes (p=0.00)	RFF-BG
Tireworld	0.91	0.71	0.68	0.21	yes (p=0.00)	yes (p=0.00)	SEH$^+(F)$
Towers of Hanoi	0.58	0.48	0.64	0.01	yes (p=0.03)	yes (p=0.00)	SEH$^+(F)$
Zenotravel	0.90	0.02	1.20	0.04	yes (p=0.00)	no (p=0.27)	SEH$^+(F)$
PI Problems	0.55	0.51	0.91	0.50	yes (p=0.00)	yes (p=0.00)	SEH$^+(F)$

Table 7: Aggregated comparison of SEH$^+(F)$ against RFF-BG. The River and Towers of Hanoi domains required extending sampling to 60 samples as per the experimental protocol described in Section 5.5.2. The values and p-values of JSLEN-RATIO and JSTIME-RATIO in Pitchcatch are not available due to the zero success ratio of RFF-BG in this category.

Category	SR of SEH(L)	SR of Greedy(L)	JSLEN-RATIO (Greedy/ SEH)	JSTIME-RATIO (Greedy/ SEH)	SR Difference Significant? (p-value)	JSLEN-RATIO Significant? (p-value)	Winner
BLOCKSWORLD	1.00	1.00	7.00	3.69	NO (p=1.00)	YES (p=0.00)	SEH(L)
BOXWORLD	0.89	0.89	5.00	0.55	NO (p=1.00)	YES (p=0.00)	SEH(L)
EXPLODING BLOCKSWORLD	0.10	0.02	1.09	1.00	YES (p=0.00)	NO (p=0.31)	SEH(L)
SYSTEMATIC -TIRE	0.34	0.14	0.75	0.39	YES (p=0.00)	YES (p=0.00)	SEH(L)
TIREWORLD	0.90	0.89	1.05	1.05	NO (p=0.92)	NO (p=0.60)	Inconclusive
TOWERS OF HANOI	0.60	0.00	–	–	YES (p=0.00)	–	SEH(L)
ZENOTRAVEL	0.58	0.03	13.25	5.66	YES (p=0.00)	YES (p=0.00)	SEH(L)

Table 8: Aggregated comparison of SEH(L) against Greedy(L). The values of JSLEN-RATIO and JSTIME-RATIO and p-value of JSLEN-RATIO in TOWERS OF HANOI are not available due to the zero success ratio of Greedy(L) in this category.

Category	SR of SEH(L)	SR of FF-Replan	JSLEN-RATIO (FFR/ SEH(L))	JSTIME-RATIO (FFR/ SEH(L))	SR Difference Significant? (p-value)	JSLEN-RATIO Significant? (p-value)	Winner
BLOCKSWORLD	1.00	0.83	0.99	2.06	YES (p=0.00)	NO (p=1.00)	SEH(L)
BOXWORLD	0.89	0.88	3.61	0.54	NO (p=0.97)	YES (p=0.00)	SEH(L)
EXPLODING BLOCKSWORLD	0.10	0.46	0.71	0.73	YES (p=0.00)	YES (p=0.00)	FF-Replan
SYSTEMATIC -TIRE	0.34	0.10	0.28	0.18	YES (p=0.00)	YES (p=0.00)	SEH(L)
TIREWORLD	0.90	0.70	0.66	0.51	YES (p=0.00)	YES (p=0.00)	SEH(L)
TOWERS OF HANOI	0.60	0.42	0.64	4.76	YES (p=0.00)	YES (p=0.00)	SEH(L)
ZENOTRAVEL	0.58	1.00	0.58	0.03	YES (p=0.00)	YES (p=0.00)	FF-Replan

Table 9: Aggregated comparison of SEH(L) against FF-Replan (FFR).

policy that greedily seeks to avoid dropping, similar to "earliest deadline first". The problems SEH encounters in our evaluation in SCHEDULE suggest future work in automatically recognizing domains where large MDP construction is proving futile and automatically reducing MDP size limits to adapt performance towards the behavior of a greedy policy. We note that across all tested benchmark domains and both heuristics, there is only one domain/heuristic combination where this phenomenon arose in practice.

6.2.2 SEH/SEH$^+$ VERSUS FF-REPLAN AND RFF-BG

We have also demonstrated performance improvement of SEH$^+(F)$ over the best performing planners in the first three international probabilistic planning competitions, outperforming FF-Replan in ten out of fifteen categories while losing in three (EXPLODING BLOCKSWORLD, PITCHCATCH, and ZENOTRAVEL), and outperforming RFF-BG in 12 out of 15 categories while losing in EXPLODING BLOCKSWORLD and SYSTEMATIC-TIRE. Additionally, SEH(L) outperforms FF-Replan in five out of seven categories while losing in EXPLODING BLOCKSWORLD and ZENOTRAVEL. In this section we discuss the categories where SEH$^+(F)$ and SEH(L) lose to FF-Replan and RFF-BG.

ZENOTRAVEL is a logistics domain where people are transported between cities via airplanes and each load/unload/fly action has a non-zero probability of having no effect. As a result, it takes an uncertain number of attempts to complete each task. In domains where the only probabilistic effect is a choice between change and no change, the all-outcome determinization leads to a "safe" determinized plan for FF-Replan–one in which no replanning is needed to reach the goal. In such domains, including ZENOTRAVEL, all-outcomes determinization can provide an effective way to employ deterministic enforced hill-climbing on the problem. We note though though, that determinization still ignores the probabilities on the action outcomes, which can lead to very bad choices in some domains (not ZENOTRAVEL). While both deterministic and stochastic enforced hill-climbing must climb out of large basins in ZENOTRAVEL, the substantial overhead of stochastic backup computations during basin expansion leads to at least a constant factor advantage for deterministic expansion. An extension to SEH that might address this problem successfully in future research would detect domains where the only stochastic choice is between change and non-change, and handle such domains with more emphasis on determinization.

EXPLODING BLOCKSWORLD is a variant of the blocks world with two new predicates **detonated** and **destroyed**. Each block can detonate once, during put-down, with some probability, destroying the object it is being placed upon. The state resulting from the action depicted in Figure 3 has a delete-relaxed path to the goal, but no actual path, so this state is a dead-end attractor for delete-relaxation heuristics such as CR-FF. FF-Replan or RFF-BG will never select this action because there is no path to the goal including this action. SEH$^+(F)$ with the weak dead-end detection used in these experiments will select the dead action shown, resulting in poor performance when this situation arises. It would be possible to use all-outcomes determinization as an improved dead-end detector in conjunction with SEH$^+(F)$ in order to avoid selecting such actions. Any such dead-end detection would have to be carefully implemented and managed to control the run-time costs incurred as SEH relies critically on being able to expand sufficiently large local MDP regions during online action selection.

In PITCHCATCH, there are unavoidable dead-end states (used by the domain designers to simulate cost penalties). However, the CR-FF heuristic, because it is based on all-outcomes determinization, assigns optimistic values that correspond to assumed avoidance of the dead-end states. As a

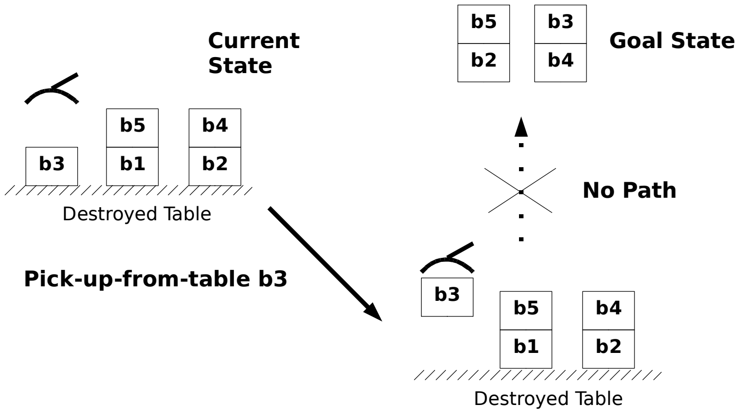


Figure 3: An illustration of a critical action choice of SEH$^+(F)$ in an EXPLODING BLOCKSWORLD problem (IPPC2 P1). The middle state has no actual path to the goal but has a delete-relaxed path to the goal. Due to the table having been exploded, no block can be placed onto the table, resulting in the middle state being a dead-end state. The middle state is a dead-end with an attractive heuristic value without regard to whether the blocks shown have remaining explosive charge or not, so this state feature is not shown.

result, local search by SEH$^+(F)$ is unable to find any expected improvement on the CR-FF values, and falls back to biased random walk in this domain. This domain suggests, as do the other domains where SEH$^+(F)$ performs weakly, that further work is needed on managing domains with unavoidable deadend states.

The two categories where SEH(L) loses to FF-Replan (EXPLODING BLOCKSWORLD and ZENOTRAVEL) are also categories where SEH$^+(F)$ loses to FF-Replan. Greedily following the learned heuristics in these two categories leads to lower success ratio than greedily following CR-FF, suggesting more significant flaws in the learned heuristics than in CR-FF. Although SEH is able to give at least a five-fold improvement over greedy following, in success ratio in these two categories, this improvement is not large enough for SEH(L) to match the performance of SEH$^+(F)$ or FF-Replan, both based on the relaxed-plan heuristic of FF.

SEH$^+$ loses to RFF in SYSTEMATIC-TIRE due to weak performance in Triangle Tireworld problems. Triangle Tireworld provides a map of connected locations arranged so that there is a single "safe" path from the source to the destination, but exponentially many shorter "unsafe" paths[8]. Determinizing heuristics do not detect the risk in the unsafe paths and so greedy following of such heuristics will lead planners (such as SEH$^+$) to take unsafe paths, lowering their success rate. While our results above show that SEH$^+$ can often repair a flawed heuristic, in the Triangle Tireworld domain the heuristic attracts SEH$^+$ to apparent improvements that are actually dead-ends.

In contrast, RFF is designed to increase robustness for determinized plans with a high probability of failure. RFF will continue planning to avoid such failure rather than relying on replanning after failure. Because the initial determinized plan has a high probability of failure (relative to RFF's

[8] The "safe" path can be drawn as following two sides of a triangular map, with many unsafe paths through the interior of the triangle. Safety in this domain is represented by the presence of spare tires to repair a flat tire that has 50% chance of occurring on every step.

Category	SR of SEH$^+(F)$	SR of FF-Replan	JSLEN-RATIO (FFR/ SEH$^+$)	JSTIME-RATIO (FFR/ SEH$^+$)	SR Difference Significant? (p-value)	JSLEN-RATIO Significant? (p-value)	Winner
BLOCKSWORLD	0.70	0.37	0.72	0.88	YES (p=0.00)	YES (p=0.00)	SEH$^+(F)$
BOXWORLD	0.67	0.34	5.02	0.98	YES (p=0.00)	YES (p=0.00)	SEH$^+(F)$

Table 10: Aggregated comparison of SEH$^+(F)$ against FF-Replan in scaled-up problems.

Category	SR of SEH$^+(F)$	SR of RFF-BG	JSLEN-RATIO (RFF-BG/ SEH$^+$)	JSTIME-RATIO (RFF-BG/ SEH$^+$)	SR Difference Significant? (p-value)	JSLEN-RATIO Significant? (p-value)	Winner
BLOCKSWORLD	0.70	0.33	0.46	0.14	YES (p=0.00)	YES (p=0.00)	SEH$^+(F)$
BOXWORLD	0.67	0.00	0.88	10.81	YES (p=0.00)	–	SEH$^+(F)$

Table 11: Aggregated comparison of SEH$^+(F)$ against RFF-BG in scaled-up problems.

threshold), RFF extends the plan before execution and can often detect the need to use the longer, safe route.

6.2.3 PERFORMANCE ON LARGE PROBLEMS

In order to demonstrate that the advantages of SEH are emphasized as problem size grows, we present aggregated performance of SEH$^+(F)$ on additional large-sized problems we have generated using generators provided by the first IPPC. As such scaling experiments are computationally very expensive, we have only run two domains that have been most widely evaluated in the planning literature: BLOCKSWORLD and BOXWORLD (which is a stochastic version of logistics). For BLOCKSWORLD, we generated 15 problems each for 25- and 30-block problems. For BOXWORLD, we generated 15 problems for the size of 20 cities and 20 boxes. (Only one problem across the three competitions reached this size in BOXWORLD, and that problem was unsolved by the competition winner, RFF.) The aggregated results against FF-Replan and RFF-BG are presented in Tables 10 and 11. The experiments for these scaled-up problems consumed 3,265 hours of CPU time and show that SEH$^+(F)$ successfully completed a majority of the attempts while FF-Replan and RFF succeeded substantially less often[9].

Note that although the FF heuristic is very good on BOXWORLD and other logistics domains, the failure of all-outcomes determinization to take into account the probabilities on action outcomes is quite damaging to FFR in BOXWORLD, leading the planner to often select an action "hoping" for its

[9] Our statistical protocol requires 30 samples of a random variable averaging performance over 5 solution attempts, for each planner for each problem. With 45 problems and 3 planners, this yields 30*5*45*3=20,250 solution attempts, each taking approximately 10 CPU minutes on these large problems.

low-probability “error outcome”. We note that RFF uses a most-probable-outcome determinization and will not suffer from the same issues as FFR in the boxworld. Given the high accuracy of the FF heuristic in the boxworld, we believe that the ideas in RFF can likely be re-implemented and/or tuned to achieve better scalability in the boxworld problems. We leave this possibility as a direction for future work on understanding the scalability of RFF.

7. Summary

We have proposed and evaluated stochastic enforced hill-climbing, a novel generalization of the deterministic enforced hill-climbing method used in the planner FF (Hoffmann & Nebel, 2001). Generalizing deterministic search for a descendant that is strictly better than the current state in heuristic value, we analyze a heuristic-based MDP around any local optimum or plateau reached at increasing horizons to seek a policy that expects to exit this MDP with a better valued state. We have demonstrated that this approach provides substantial improvement over greedy hill-climbing for heuristics created using two different styles for heuristic definition. We have also demonstrated that one resulting planner is a substantial improvement over FF-Replan (Yoon et al., 2007) and RFF (Teichteil-Konigsbuch et al., 2010) in our experiments.

We find that the runtime of stochastic enforced hill-climbing can be a concern in some domains. One reason for the long runtime is that the number and size of local optima basins or plateaus may be large. Currently, long runtime is managed primarily by reducing to biased random walk when resource consumption exceeds user-set thresholds. A possible future research direction regarding this issue is how to prune the search space automatically by state or action pruning.

Acknowledgments

This material is based upon work supported in part by the National Science Foundation, United States under Grant No. 0905372 and by the National Science Council, Republic of China (98-2811-M-001-149 and 99-2811-M-001-067).

References

Aarts, E., & Lenstra, J. (Eds.). (1997). *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc.

Barto, A. G., Bradtke, S. J., & Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, *72*, 81–138.

Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*. Athena Scientific.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Bonet, B., & Geffner, H. (2005). mGPT: A probabilistic planner based on heuristic search. *Journal of Artificial Intelligence Research*, *24*, 933–944.

Bonet, B., & Geffner, H. (2006). Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings, and its application to MDPs. In *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling*, pp. 142–151.

Bryce, D., & Buffet, O. (2008). International planning competition uncertainty part: Benchmarks and results.. http://ippc-2008.loria.fr/wiki/images/0/03/Results.pdf.

Buffet, O. (2011) Personal communication.

Cerny, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *J. Optim. Theory Appl.*, *45*, 41–51.

Dean, T., Kaelbling, L. P., Kirman, J., & Nicholson, A. (1995). Planning under time constraints in stochastic domains. *Artificial Intelligence*, *76*, 35–74.

Domshlak, C., & Hoffmann, J. (2007). Probabilistic planning via heuristic forward search and weighted model counting. *Journal of Artificial Intelligence Research*, *30*, 565–620.

Fahlman, S., & Lebiere, C. (1990). The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2*, pp. 524 – 532.

Gardiol, N. H., & Kaelbling, L. P. (2003). Envelope-based planning in relational MDPs. In *Proceedings of the Seventeenth Annual Conference on Advances in Neural Information Processing Systems*.

Gerevini, A., & Serina, I. (2003). Planning as propositional CSP: from Walksat to local search techniques for action graphs. *Constraints*, *8*(4), 389–413.

Gordon, G. (1995). Stable function approximation in dynamic programming. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 261–268.

Hansen, E., & Zilberstein, S. (2001). LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, *129*, 35–62.

Hoffmann, J. (2003). The Metric-FF planning system: Translating “ignoring delete lists” to numeric state variables. *Journal of Artificial Intelligence Research*, *20*, 291–341.

Hoffmann, J., & Brafman, R. (2005). Contingent planning via heuristic forward search with implicit belief states. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling*.

Hoffmann, J., & Brafman, R. (2006). Conformant planning via heuristic forward search: A new approach. *Artificial Intelligence*, *170*(6-7), 507 – 541.

Hoffmann, J. (2005). Where ‘ignoring delete lists’ works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research*, *24*, 685–758.

Hoffmann, J., & Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, *14*, 253–302.

Kautz, H., & Selman, B. (1992). Planning as satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI’92)*.

Kirkpatrick, S., Gelatt, Jr, C., & Vecchi, M. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Little, I., & Thiebaux, S. (2007). Probabilistic planning vs replanning. In *Workshop on International Planning Competition: Past, Present and Future (ICAPS)*.

Liu, C., & Layland, J. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association for Computing Machinery*, *20*, 46–61.

Mahadevan, S., & Maggioni, M. (2007). Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, *8*, 2169–2231.

Nilsson, N. (1980). *Principles of Artificial Intelligence*. Tioga Publishing, Palo Alto, CA.

Puterman, M. L. (2005). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

Sanner, S., & Boutilier, C. (2009). Practical solution techniques for first-order MDPs. *Artificial Intelligence*, *173*(5-6), 748–788.

Selman, B., Kautz, H., & Cohen, B. (1993). Local search strategies for satisfiability testing. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 521–532.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Teichteil-Konigsbuch, F., Kuter, U., & Infantes, G. (2010). Incremental plan aggregation for generating policies in MDPs. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pp. 1231–1238.

Tesauro, G., & Galperin, G. (1996). On-line policy improvement using Monte-Carlo search. In *NIPS*.

Wu, J., & Givan, R. (2007). Discovering relational domain features for probabilistic planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, pp. 344–351.

Wu, J., & Givan, R. (2010). Automatic induction of Bellman-Error features for probabilistic planning. *Journal of Artificial Intelligence Research*, *38*, 687–755.

Yoon, S., Fern, A., & Givan, R. (2007). FF-Replan: A baseline for probabilistic planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, pp. 352–358.

Younes, H., Littman, M., Weissman, D., & Asmuth, J. (2005). The first probabilistic track of the international planning competition. *Journal of Artificial Intelligence Research*, *24*, 851–887.