Stochastic Enforced Hill-Climbing

Jia-Hong Wu and Rajesh Kalyanam and Robert Givan
Electrical and Computer Engineering, Purdue University, W. Lafayette, IN 47907
{*jw, rkalyana, givan*}*@purdue.edu*

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 local Markov-decision process model of the basin in order to find a good escape policy exiting the local optimum.

We evaluate our proposal in a wide range of recent probabilistic planning-competition benchmark domains. For evaluation, we show that stochastic enforced hill-climbing produces better policies than greedy action choice for value functions derived in two very different ways. First, we propose a novel heuristic function derived from the ideas in the effective re-planner FF-Replan. This new "controlled-randomness FF heuristic" is the deterministic FF heuristic computed on the simple determinization of the probabilistic problem that makes available a deterministic transition wherever a probabilistic transition was possible. Our results show that stochastic enforced hill-climbing with this heuristic significantly outperforms simply greedily following the heuristic, and also substantially outperforms FF-Replan. We additionally evaluate our technique on automatically learned value functions that on their own perform at the state-of-the-art when used to construct a greedy policy, and again show significant improvement over greedy action selection.

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, and 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 and Nebel 2001)

The immediate goal of enforced hill-climbing is to find a descendant that is strictly better than the current state in heuristic value by performing a local search. The planner then moves to that descendant and repeats this process. The effectiveness of enforced hill-climbing is demonstrated by the success of the planner FF (Hoffmann and Nebel 2001).

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, and Givan 2007)—a current top performer for 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 single action, for the current state. After this action is committed to and executed, 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.

We propose a novel tool for stochastic planning by generalizing enforced hill-climbing to stochastic domains. Rather than seeking a sequence of actions deterministically leading to a better state, our method constructs a breadth-first local Markov Decision Process (MDP) around the current state and seeks a *policy* that expects to exit this MDP with a better valued state. When such a policy is found, the method executes the policy until the local MDP is exited. Critical to this process is the direct incorporation of the probabilistic model in finding the desired policy. Here, we find this policy using value iteration on the local MDP, where the only rewards are the heuristic values assigned at the exits. As in enforced hill-climbing, breadth-first expansion around a state occurs only when there is no action available with Q-value (relative to the heuristic) achieving the current state's 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 size of the local MDPs needed to find expected heuristic improvement.

Stochastic enforced hill-climbing can be applied to any heuristic function. We test the technique on a broad range of domains from the first two international probabilistic planning competitions. We select two very different heuristic functions: First we use a novel heuristic function based on the determinization from the successful re-planner FF-Replan. This new "controlled-randomness FF heuristic" is the FF heuristic (Hoffmann and Nebel 2001) computed on the FF-Replan determinization of the probabilistic problem. Our results show that stochastic enforced hill-climbing with this heuristic outperforms FF-Replan substantially, and also outperforms simply greedily following the heuristic. We additionally evaluate our technique on value functions produced by a relational feature-learning method in (Wu and Givan 2007). These value functions have been shown to perform at the state-of-the-art when used to construct a greedy policy. We again show improvement over greedy selection when using these value functions as the heuristic.

Technical Background and Related Work

Goal-oriented Markov decision processes We give a brief review of Markov decision processes (MDPs) specialized to goal-region objectives. For more detail, see (Bertsekas and Tsitsiklis 1996) and (Sutton and Barto 1998).

A stochastic shortest path Markov decision process (MDP) M is a tuple (S, A, R, T) with finite state and action spaces S and A, reward function $R : S \times A \times S \to \mathbb{R}$, and a transition probability function $T : S \times A \to \mathcal{P}(S)$ that maps (state, action) pairs to probability distributions over S. To avoid discounting while still assigning finite value to each state, we require that S contain a zero-reward absorbing state $\perp$, i.e., such that $R(\perp, a, s) = 0$ and $T(\perp, a, \perp) = 1$ for all $s \in S$ and $a \in A$, and that there is some policy (as defined below) from any state leading to $\perp$ with probability 1.

Goal-oriented MDPs are MDPs where there is a subset $G \subseteq S$ of the statespace, containing $\perp$, such that: (1) $R(g, a, s')$ is zero whenever $g \in G$ and minus one otherwise, and (2) $T(g, a, \perp)$ is one for all $a \in A$ and $g \in G$.

Given *policy* $\pi : S \to A$ for an MDP, the value function $V^{\pi}(s)$ gives the expected cumulative reward obtained from state s selecting action $\pi(s)$ at each state encountered[1]. There is at least one optimal policy π^* for which $V^{\pi^*}(s)$, abbreviated $V^*(s)$, is no less than $V^{\pi}(s)$ at every state s, for any other policy π. The following "Q function" evaluates an action a with respect to a value function V by using V to estimate the value after action a is applied,

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

Recursive Bellman equations use $Q()$ to describe V^* and V^{π} as follows. First, $V^{\pi}(s) = Q(s, \pi(s), V^{\pi})$. Then, $V^*(s) = \max_{a \in A} Q(s, a, V^*)$. Also using $Q()$, we can select an action greedily relative to any value function. The policy Greedy(V) selects, at any state s, a randomly selected action from $\arg\max_{a \in A} Q(s, a, V)$.

Value iteration iterates the operation $V'(s) = \max_{a \in A} Q(s, a, V)$, computing the "Bellman update" V' from V, producing a sequence of value functions converging to V^*, regardless of the initial V used, as shown by Bertsekas for a similar formulation in (Bertsekas 1995).

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 et al. 2005), as used in our experiments. Our technique below avoids converting the PPDDL problem explicitly into the above form, but constructs a sequence of smaller problems of this form modeling heuristic flaws.

Local Markov decision processes Given a heuristic function $h : S \to \mathbb{R}$ estimating V^* for a MDP $M = (S, A, R, T)$, i.e., estimating negated distance to the absorbing state, we define the sub-MDP induced by state set $S' \subseteq S$ to be the MDP that equals M inside S', but gives terminal reward according to h upon exiting S'. Formally, the sub-MDP induced by S' and h is given by (S, A, R', T'), where R' and T' are given as follows. Let s be any state in S and a any action in A. For any state $s' \in S'$, we take $R'(s', a, s) = 0$ and $T'(s', a, s) = T(s', a, s)$. For any state x in $S - S'$, $R'(x, a, s) = h(x)$ and $T'(x, a, \perp) = 1$. We refer to the number of states reachable from S' in zero or one steps as the "size of the sub-MDP induced by S'".

The rough motivation for setting action rewards to zero for the analysis within S' is that such actions are being considered by our method to remediate a flawed heuristic; the cumulative negative action reward paid to reach a state of higher heuristic value is a measure of the magnitude of the flaw in the heuristic, but we remove this reward from the analysis in order to directly express the subgoal of "reaching a state with higher heuristic value". Instead of diluting this subgoal by adding in negative action rewards, our methods seek the "shortest" path to a heuristic improvement by enlarging S' in a breadth first manner. The resulting breadth-first expansion of S' generalizes if needed to uniform-cost expansion to handle more general action costs.[2]

[1]Formally, we have restricted consideration to domains where some policy reaches a goal state with probability one from all states, to avoid discounting. In practice we can and do handle deadend states by assigning them large negative values if they are recognized by simple reachability tests, and by adjusting the parameters for our algorithm as discussed in subsection "Deadend handling." We defer more formal algorithmic treatment of deadend domains for future work, but note that some domains evaluated in this paper do contain unavoidable deadends.

[2]Also note that in this zero-reward undiscounted setting, greedily acting based upon the converged value function is not guaranteed to achieve that value. It is necessary to remember the action

Note: technically, we seek to "climb" in "value", even though we think of heuristics as measuring distance to the goal. We simply use the negative value of the distance-to-go heuristic as our value function.

Related work Stochastic enforced hill-climbing dynamically constructs local MDPs to find a local policy leading to heuristically better state regions. The concept of forming sub-MDPs, or "envelopes", and using them to facilitate probabilistic planning has been used in previous research such as (Bonet and Geffner 2006; Dean et al. 1995), which we briefly review here.

The envelope-based method in (Dean et al. 1995) starts with a partial policy in a restricted area of the problem (the "envelope"), then iteratively improves the solution quality by changing the envelope and recomputing the partial policy. The typical assumption of implementing this method is to have an initial trajectory from the starting state to the goal, generated by some stochastic planner.

Another line of work (Barto, Bradtke, and Singh 1995; Hansen and Zilberstein 2001; Bonet and Geffner 2006) starts from only the initial state in an envelope, iteratively expands the envelope by expanding states according to state values and using dynamic programming methods to backup state values, 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. Preliminary experiments with one such recent previous technique, LDFS (Bonet and Geffner 2006), on the domains and heuristics used for evaluation in this paper gave poor results. 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 V^* as input) and possibly vastly many others as well during the computation.

Stochastic Enforced Hill Climbing

Deterministic enforced hill-climbing (Hoffmann and Nebel 2001) searches for a strictly improving successor state and returns a path from the initial state to such a successor. This path is an action sequence that guarantees reaching the desired successor. In a stochastic environment, there may be no single improved descendant that can be reached with probability one, as the actions may have multiple stochastic outcomes. Thus, in stochastic enforced hill-climbing, we generalize the idea of searching for a single strictly improved successor state to finding a policy that expects to improve on the heuristic value of the initial state inside a dynamically constructed sub-MDP. As formalized in "Local Markov decision processes" above, the sub-MDP ignores the negative action rewards paid to reach such improved states; the secondary goal of minimizing these costs is embodied by constructing the smallest sub-MDP enabling success, as described formally below. Note that, in contrast to replanning techniques, with this approach the system adjusts to the uncertainty in action outcomes without replanning.

We present pseudo-code for stochastic enforced hill-climbing in Figure 1, and explain the terminology used in the pseudo-code next. The algorithm assumes a non-positive heuristic function $h : S \to \mathbb{R}$ as input that assigns zero to all goal states. Stochastic enforced hill-climbing iteratively builds and solves sub-MDPs and seeks improved policies inside such sub-MDPs[3]. Each sub-MDP is induced as defined previously by a state set Γ together with the heuristic function h. We use two parameters k and α to control the aggressiveness with which the sub-MDPs are constructed in an attempt to escape the local optimum or plateau. The horizon parameter k limits the number of steps from the entry state s_0; the heuristic radius parameter α limits the heuristic distance from $h(s_0)$ that states in the sub-MDP may have, representing the energy barrier to escape. We assume some schedule for selecting more and more aggressive values of k and α to construct increasingly large sub-MDP problems seeking an exit. Our algorithm applies to any such schedule.

For any pair (k, α) in the schedule, we define a state set $\Gamma_{k\alpha}$. For any α, we define $\Gamma_{0\alpha}(s_0) = \{s_0\}$. We define an operation Extend on a state set to be Extend$(\Gamma) = \{s' \mid \exists s \in \Gamma, \exists a \in A, P(s, a, s') > 0\}$. We can then iteratively extend $\Gamma_{k\alpha}$ to be $\Gamma_{(k+1)\alpha}(s_0) = \Gamma_{k\alpha} \cup \{s' \in Extend(\Gamma_{k\alpha}) \mid |h(s') - h(s_0)| \leq \alpha\}$.

Thus, in line 6 of Figure 1 , value iteration is conducted for increasingly large sub-MDPs around s_0, seeking a policy improving $h(s_0)$[4]. Depending on the schedule chosen for k and α, implementation choices may allow reuse of statespace and value information as larger MDPs are constructed, expanding on smaller ones. The implementation may also proceed down the schedule without value iteration when no new states of high value are added to the sub-MDP.

Early termination The primary termination condition for repeated sub-MDP construction is the discovery of a policy improving on the heuristic estimate of the initial state. However, for flawed heuristic functions that overestimate state value, especially in domains with unavoidable deadends, this may not be possible. For this reason, in practice, the algorithm stops enlarging sub-MDPs when user-specified resource limits are exceeded.

which *first* achieved the converged value during value iteration, for each state, and take that action in preference to others that may look equal in greedy value due to zero-cost cycles.

[3]Note that we must avoid building the full MDP explicitly as it is exponentially large.

[4]Given the negative-reward-per-step structure of a goal-oriented MDP, $h()$ is an estimate of (negated) expected distance to the goal, so this subgoal corresponds to seeking a policy that expects to move closer to the goal.

```
Stochastic enforced hill-climbing
1.  Repeat
2.     // for some schedule of $(k, \alpha)$ pairs $(k_i, \alpha_i)$
3.     $s_0 \leftarrow$ current state
4.     $i = 1, \Gamma = \Gamma_{k_1\alpha_1}$
5.     Repeat
6.        $V \leftarrow$ Value iteration in sub-MDP($\Gamma$,$h$)
7.        $i = i + 1, \Gamma = \Gamma_{k_i\alpha_i}$
8.     Until $V(s_0) > h(s_0)$ or resource consumption exceeds
       user-set limits
9.     Follow Greedy($V$) until exit $\Gamma$ or a state visited $\sigma$ times
10.    If $h$(new current state) $< h(s_0)$, take random walk of
       $\omega$ steps
11. Until goal achieved
```

Figure 1: *Pseudo-code for stochastic enforced hill-climbing.*

Once a sub-MDP is constructed assigning $V(s_0) > h(s_0)$ or exceeding the resource limits, the system executes the greedy policy within the sub-MDP until the policy exits the sub-MDP. Again, in practice, we impose a bound σ on the number of times a state may be repeated during the execution. If the greedy policy execution terminates in a state with poor heuristic value (worse than $h(s_0)$), our method adds a random walk of length ω; this additional 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.

Deadend handling The implementation tested in this paper adds ad-hoc handling of dead-end states, which can be incompletely identified by the heuristics we use. Recognized dead-ends are assigned a large negative value τ. Also, once a dead-end state is detected in a problem, we set a minimum sub-MDP size of 500 states for every sub-MDP constructed for that problem (to reduce the chance of greedily falling into a dead-end). Also, because dead-ends break our termination guarantee based on improving on $h(s_0)$, for any sub-MDP that contains a known dead-end state, we terminate sub-MDP growth once a good exit is discovered (with h greater than $h(s_0)$) and the sub-MDP size is at least 2000.

Setup for Empirical Evaluation

Here, we prepare for our results section by describing the parameters used for our method, the heuristics we will test the method on, the domains in which the tests will be conducted, and issues arising in interpreting the results.

Parameterizing the algorithm When implementing stochastic enforced hill-climbing, we use a schedule of choices for the horizon k and heuristic radius α parameters that explores a sequence of more and more aggressive heuristic radius values $\alpha_1, \alpha_2, \ldots$. For each radius parameter α_i, a sequence of horizons k is considered $0, 1, \ldots, 10i$, before extending the radius. We take α_1 to be zero and define subsequent α_{i+1} as follows. Let y_i be the smallest heuristic radius value such that Γ_{ky_i} properly contains $\Gamma_{k\alpha_i}$ for any k in $0, 1, \ldots, 10i$. We then choose $\alpha_{i+1} = 1.5 * y_i$. If there is no such y_i, stochastic enforced hill-climbing cannot proceed further and the schedule terminates. The value y_i is easily computed during sub-MDP construction using α_i. Throughout our experiments, we terminate sub-MDP construction (as in line 8 of table 1) when the size of the sub-MDP exceeds $1.5 * 10^5$ or the runtime to construct the sub-MDP exceeds 1 minute. We set the limit on repeated states during policy execution σ to 50 and the random walk length ω to 9. The value τ assigned to recognized dead-end states is set to $-1.0 * 10^5$. We denote this implementation running on heuristic h with **SEH**(h).

Goals of the evaluation Our empirical goal is to show that stochastic enforced hill-climbing improves significantly upon greedy following of the same heuristic (using the policy Greedy(h) as described in the technical background above. We will show this is true both for a simple new heuristic based on determinization, described below, and for learned heuristics from previously published research. A secondary goal of our evaluation is to show that the result is strong in comparison to FF-Replan.

The Controlled-randomness FF heuristic We define a novel heuristic function, the "controlled-randomness FF heuristic" (CR-FF), as the FF heuristic (Hoffmann and Nebel 2001) computed on the FF-Replan (Yoon, Fern, and Givan 2007) determinization of the probabilistic problem[5]. 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 such outcome happening. This effectively allows the planner to control the randomness in executing actions, making this determinization a kind of relaxation of the problem. We use the CR-FF heuristic in each domain as one kind of heuristic function h in our experiments.

For our experiments with the CR-FF heuristic, we compute the heuristic value $h(s)$ as $h(s) = -h_d(s)$ where $h_d(s)$ is the FF distance estimate (Hoffmann and Nebel 2001) of s computed on the above determinization. We denote the resulting heuristic F.

Learned heuristics We also test stochastic enforced hill-climbing on automatically generated value functions from (Wu and Givan 2007), which on their own perform at the state-of-the-art when used to construct a greedy policy. We scale these value functions to fit the non-positive range requirement of h discussed previously. These learned value functions are currently available for only five of our test domains, and so are only tested in those domains.

We note that these heuristics were learned for a discounted setting without action costs and so are not a direct fit to the negated "distance-to-go" formalization adopted here. We are still able to get significant improvements from applying our technique. We denote the resulting heuristic L.

Domains considered We evaluate stochastic enforced hill-climbing on a wide range of recent probabilistic

[5]The deterministic FF heuristic, described in (Hoffmann and Nebel 2001), from FF planner version 2.3 available at http://members.deri.at/~joergh/ff.html, computes a sequential relaxed-plan length using a sequence of ordered subgoal steps.

	SEH(F)	**Greedy**(F)	**SEH**(L)	**Greedy**(L)	**FF-R**
Avg1	0.947	0.364	0.857	0.760	0.828
Avg2	0.882	0.417	–	–	0.785

Figure 2: *Average of average success ratios across testing domains.* ***Avg1*** *represents the average across the five domains that all planners address, and* ***Avg2*** *represents the average across all nine domains for the indicated planners.* ***FF-R*** *abbreviates "FF-Replan."*

planning-competition benchmark domains. For each problem size tested in each domain, 30 problems are generated from the given problem generator and results are averaged over 30 attempts per problem. We use a length cutoff of 2000 for each attempt. Each attempt is given a time limit of 30 minutes. We run our experiments on Intel Pentium 4 Xeon 2.8GHz machines.

In (Wu and Givan 2007), value functions are learned for five domains:

- **blocksworld** and **boxworld** from the first international probabilistic planning competition (IPPC);
- **tireworld**, **zenotravel**, and **exploding blocksworld** from the second IPPC.

We use these five domains when evaluating both the CR-FF heuristic and the learned value-function heuristic. We additionally evaluate the following domains from the second IPPC using CR-FF heuristic: **Random**, **schedule**, **elevators**, and **pitchcatch**. For these four domains we take the problems from the competition rather than use the problem generators to generate new ones[6].

For each problem in each domain we show three types of measurement: success ratio (SR), average plan length over only the successful attempts (Slen), and average runtime for only the successful attempts (Stime). In **blocksworld**, **boxworld**, **tireworld**, **zenotravel**, and **exploding blocksworld**, these per-problem results are aggregated by problem size to compress the data. In addition, averaging across all problems in the domain, we also plot the overall success ratio, overall average successful plan length, and overall average runtime of successful attempts (labeled "A" on the x-axis).

Notes on interpreting the data It is very important to note that successful plan length and successful runtime are only appropriate to compare between planners when those planners exhibit very similar success ratios (ideally near 1.0). Otherwise, differences in plan lengths may be due to differences in the problems being solved rather than plan quality. Plan lengths are shown in these cases only for very coarse comparison. Some advantages for the **SEH** algorithm occur in plan length and some in success ratio, mandating careful data interpretation. Note: all runtime and plan length results are presented on a logarithmic scale.

Empirical Results

Results summary For each heuristic function h evaluated, we compare the greedy hill-climbing algorithm **Greedy**(h)

[6]We do this because for three of these "domains", the problem generators change the action set between instances, so reusing the actual instances aids in consistency with the competition.

	SEH(F)	**Greedy**(F)	**SEH**(L)	**Greedy**(L)	**FF-R**
Avg1	1.598	2.60	1.525	8.180	2.494
Avg2	1.465	2.069	–	–	1.832

Figure 3: *Average of "normalized average plan length" across domains, where plan length is normalized by dividing by the best average plan length achieved by any planner in the domain in question. We omit two domains (tireworld and exploding blocksworld) where large variations in success rate makes successful length particularly misleading.* ***Avg1*** *represents the average across the three domains that all planners address, and* ***Avg2*** *represents the average across all seven domains for the indicated planners.* ***FF-R*** *abbreviates "FF-Replan."*

and our stochastic-enforced-hillclimbing algorithm **SEH**(h) to determine whether stochastic enforced hill-climbing improves performance. We also show, on every plot, the performance of a recent stochastic re-planner, FF-Replan, that performs quite well in these benchmark domains.

In the first set of experiments, we evaluate the novel heuristic F defined above (in the section titled "The controlled-randomness FF heuristic"). In the results in the next section, we show that **SEH**(F) significantly outperforms **Greedy**(F) across seven of nine benchmark planning domains, roughly equalling **Greedy**(F) in the other two domains (tireworld and schedule).

In addition to nearly always significantly improving on greedy hill-climbing, **SEH**(F) manages very similar performance to **FF-Replan** (with small variations either way) in three of the nine domains tested, and more dominant performance in either success ratio or successful plan length in the remaining six, making it a state-of-the-art approach to these competition benchmark planning domains.

In a second set of experiments, we consider the learned heuristic functions. Here, again, **SEH**(L) significantly improves upon **Greedy**(L) in four of five domains, and slightly improves **Greedy**(L) in the other domain (tireworld). **SEH**(L) also outperforms **FF-Replan** in three of five domains, but loses in zenotravel and exploding blocksworld, where L itself is poor.

Because there is a lot of data shown across the 480 problem instances considered, we attempt in Figs. 2 and 3 a very rough aggregation for concise comparison purposes. We report the average success ratio for each of the five approaches averaged across across all domains considered, giving equal weight to each domain (combining all problem sizes within each domain). Likewise, we report the average across domains of "normalized average plan length", where plan length is normalized by dividing by the best average plan length achieved by any planner in the domain in question. For the plan length average, we omit two domains (tireworld and exploding blocksworld) where large variations in success rate, even on single problems, makes successful length particularly misleading.

On these averages, **SEH**(F) is substantially better than **FF-Replan** in both success ratio and average length. In addition, **SEH**(F) has a dramatically higher success rate than **Greedy**(F). For learned heuristics, **SEH**(L) gives a marked improvement over **Greedy**(L) in overall average success ra-

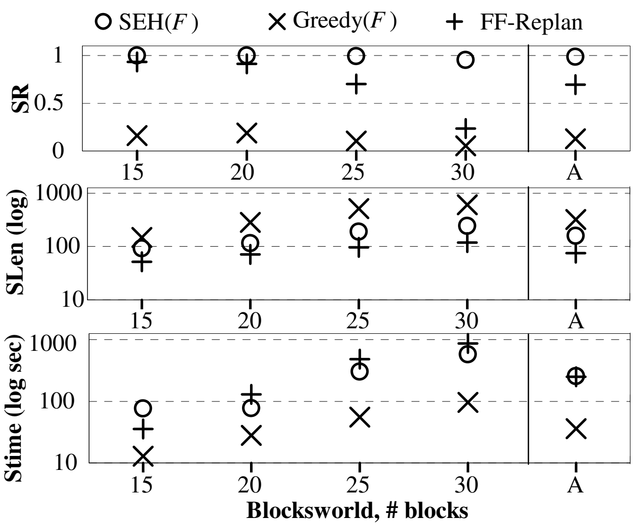

Figure 4: ***Blocksworld** results for the CR-FF heuristic.*

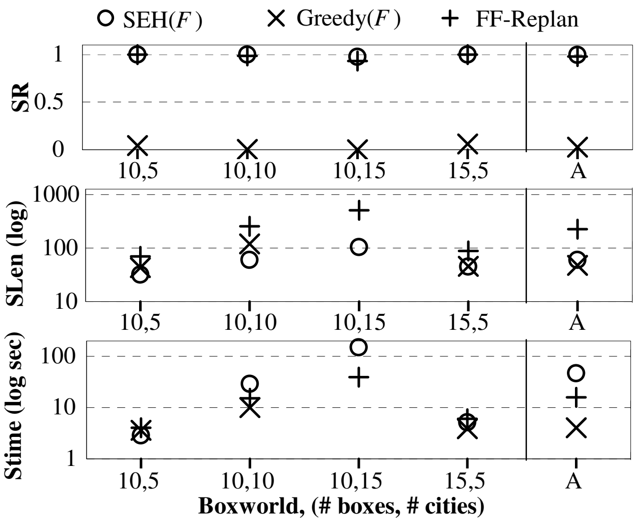

Figure 5: ***Boxworld** results for the CR-FF heuristic.*

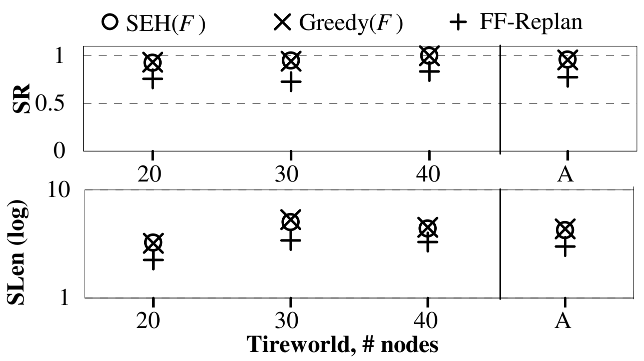

Figure 6: ***Tireworld** results for the CR-FF heuristic. Runtime results are omitted because all planners are completing each attempt with an average of less than one second.*

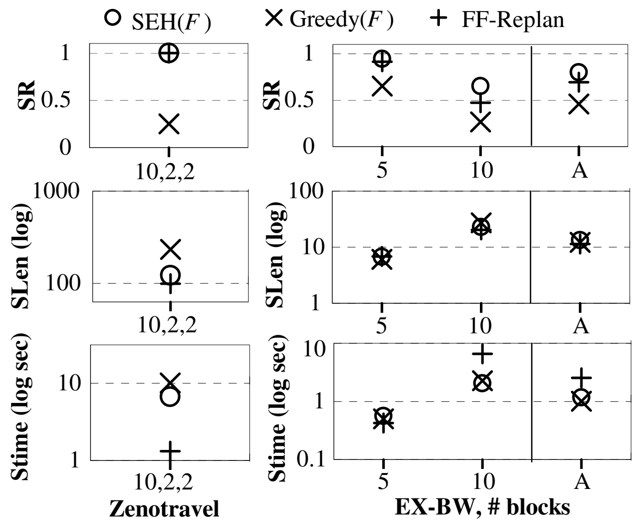

Figure 7: ***Zenotravel** and **exploding blocksworld** results for the CR-FF heuristic. The problem size for **zenotravel** is 10 cities, 2 people, 2 aircraft.*

tio, and a dramatic improvement in the average length measure. **SEH(L)** also shows similar success ratio and much shorter average length than **FF-Replan**.

Detailed results for the controlled FF heuristic In Figs. 4 to 7, we present detailed results for the CR-FF heuristic for the five domains handled by all planners.

When compared to Greedy(F), SEH(F) has significantly higher success ratio in **blocksworld**, **boxworld**, **zenotravel**, and **exploding blocksworld**. In **tireworld** SEH(F) and Greedy(F) are equal in success ratio.

When compared to FF-Replan, SEH(F) has significantly higher success ratio in **blocksworld**, **tireworld**, and **exploding blocksworld**. Note that these differences in success rate make successful length comparisons misleading in these domains. In **boxworld**, SEH(F) and FF-Replan have similar success ratio, but SEH(F) yields a significantly shorter average successful plan length than FF-Replan. In **zenotravel**, both planners are equally successful in finding the goals while FF-Replan is slightly shorter in plan length.

In Figs. 8 to 11, we present detailed results for the CR-FF heuristic for the four remaining domains, and we discuss each briefly here. For these plots, each label on the x-axis indicates the problem number from the second IPPC.

When compared to Greedy(F), SEH(F) has significantly higher success ratio in **random**, **elevators**, and **pitchcatch**, though the advantage in **random** is present on only one of fifteen problems. In **schedule**, SEH(F) and Greedy(F) exhibit similar success ratios.

When compared to FF-Replan, SEH(F) performs similarly in success ratio across two of the four domains (**random** and **elevators**), and improves on FF-Replan in the two others (**schedule** and **pitchcatch**). Also, SEH(F) finds significantly shorter plans than FF-Replan in **pitchcatch**.

Detailed results for the learned heuristics In Figs. 12 to 15, we present detailed results for the learned heuristics taken from (Wu and Givan 2007) for the five domains where learned heuristics are available, and we discuss each briefly

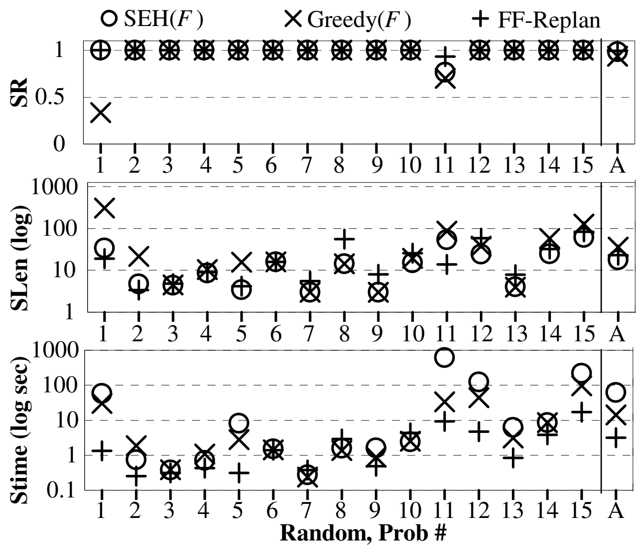

Figure 8: ***Random** results for the CR-FF heuristic.*

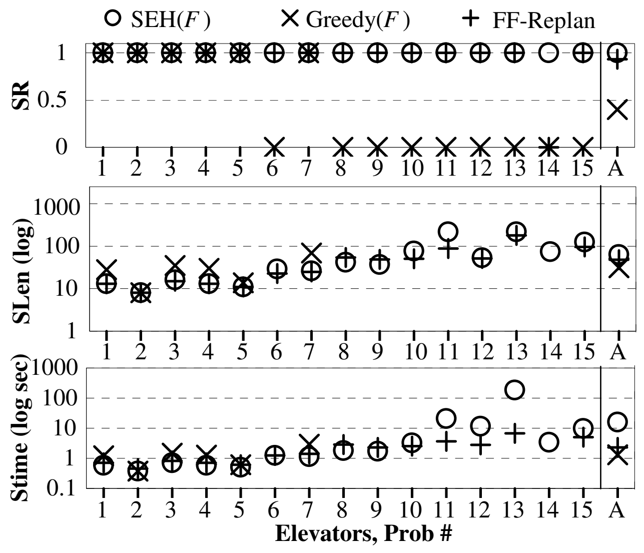

Figure 9: ***Elevators** results for the CR-FF heuristic.*

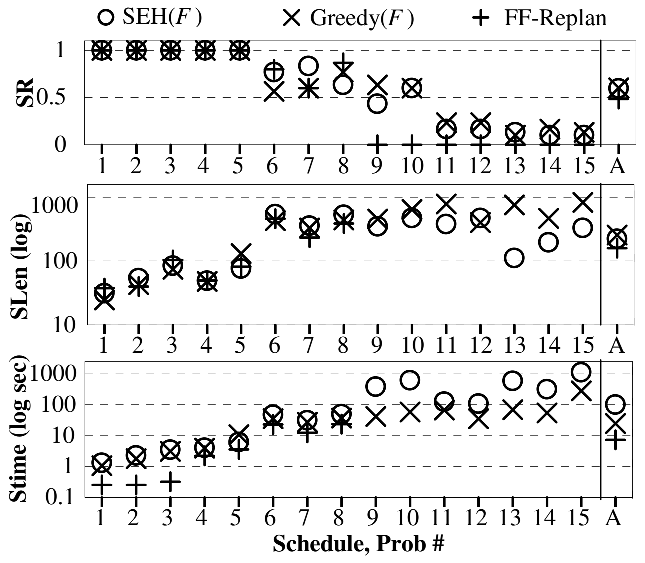

Figure 10: ***Schedule** results for the CR-FF heuristic.*

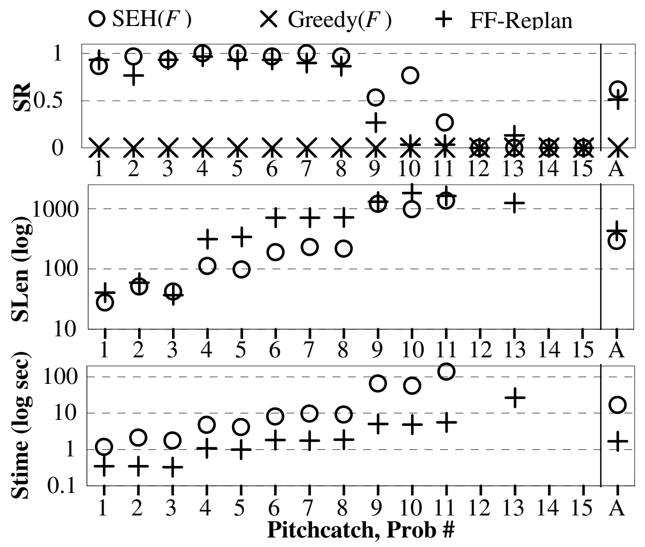

Figure 11: ***Pitchcatch** results for the CR-FF heuristic.*

here[7].

In **blocksworld** and **boxworld**, SEH(L) preserves the high success ratio of Greedy(L) and significantly improves the average successful plan length. SEH(L) improves slightly on the success ratio of Greedy(L) in **tireworld**. Note that in these three domains SEH(L) also significantly outperforms FF-Replan as Greedy(L) already has excellent performance in these domains.

In **zenotravel**, SEH(L) again has significant improvement on average successful plan length over Greedy(L). In **exploding blocksworld**, the base heuristic Greedy(L) performs so poorly as to provide inadequate guidance to SEH, although SEH(L) is able to improve on Greedy(L) in 5 blocks problems. In both of these domains, FF-Replan outperforms either hill-climbing method.

[7]We take the best-performing value functions used for comparison purpose in that paper.

Summary

We have proposed and evaluated stochastic enforced hill-climbing, a novel generalization of the deterministic enforced hill-climbing from FF (Hoffmann and Nebel 2001). Instead of finding a descendant that is strictly better than the current state in heuristic value, we construct a local MDP around any local optimum or plateau reached and 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 fourteen different heuristics in nine different domains, created using two different styles for heuristic definition. We have also demonstrated the resulting planners are a substantial improvement over FF-Replan (Yoon, Fern, and Givan 2007) in our experiments.

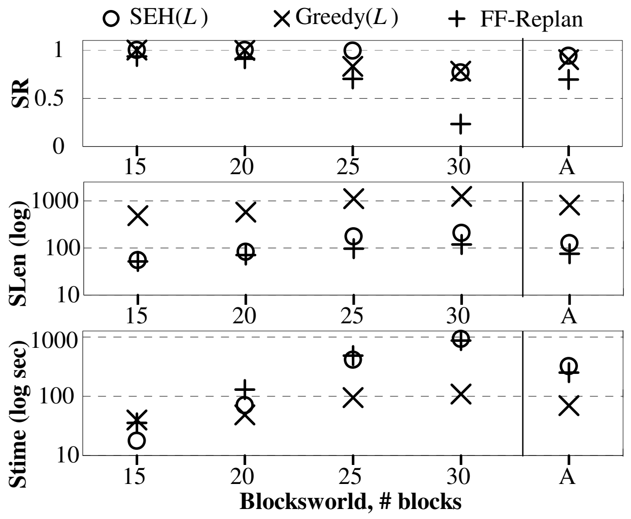

Figure 12: ***Blocksworld** results for learned value functions.*

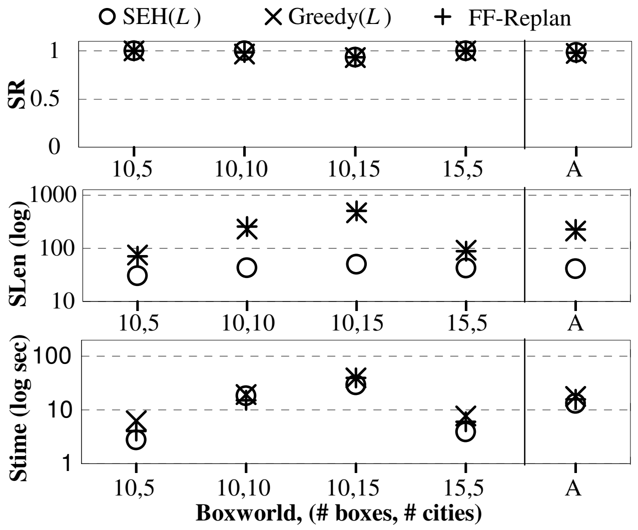

Figure 13: ***Boxworld** results for learned value functions.*

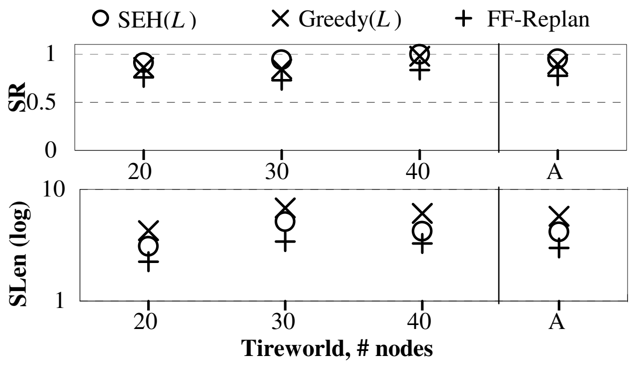

Figure 14: ***Tireworld** results for learned value functions. We omit runtime results as on average each planner uses less than 1 second per attempt.*

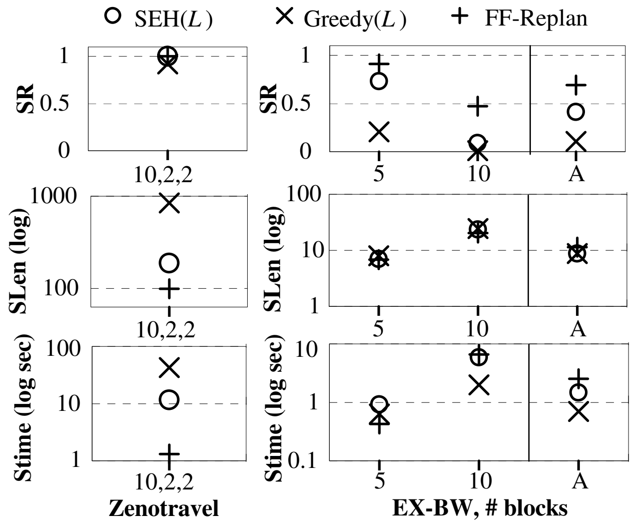

Figure 15: ***Zenotravel** and **exploding blocksworld** results for learned value functions. The problem size for **zenotravel** is 10 cities, 2 people, 2 aircraft.*

References

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1995. Learning to act using real-time dynamic programming. *AIJ* 72:81–138.

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

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

Bonet, B., and 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 *ICAPS*.

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.; and Nicholson, A. 1995. Planning under time constraints in stochastic domains. *AIJ* 76:35–74.

Hansen, E., and Zilberstein, S. 2001. LAO*: A heuristic search algorithm that finds solutions with loops. *AIJ* 129:35–62.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

Kirkpatrick, S.; Gelatt, Jr, C.; and Vecchi, M. 1983. Optimization by simulated annealing. *Science* 220:671–680.

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

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

Wu, J., and Givan, R. 2007. Discovering relational domain features for probabilistic planning. In *ICAPS*.

Yoon, S.; Fern, A.; and Givan, R. 2007. FF-Replan: A baseline for probabilistic planning. In *ICAPS*.

Younes, H.; Littman, M.; Weissman, D.; and Asmuth, J. 2005. The first probabilistic track of the international planning competition. *JAIR* 24:851–887.