Augmenting Exploration with Locally Greedy Probes

Dawson Tomasz, Richard Valenzano

Toronto Metropolitan University
dawson.tomasz@proton.me, rick.valenzano@torontomu.ca

Abstract

Enhancing Greedy Best First Search (GBFS) with stochastic exploration will often greatly improve search performance. In this work, we show that one way exploration does so is by helping the search find states that are "easy" for standard GBFS without exploration. In particular, we show that in problems in which standard GBFS struggles and exploration helps, there are often many states that are reachable from the initial state that standard GBFS can quickly find solutions from. Many such states are actually outside the Bench Transition System (BTS) — which is a structure that contains all states that standard GBFS may encounter — meaning GBFS cannot reach them without using exploration. To allow exploration mechanisms to better exploit the existence of such states, we introduce a method called locally greedy probes. Upon a successor having an improved heuristic from its parent, locally greedy probes pause exploration and greedily hill-climb along a single path as long as heuristic improvements keep occurring. Our empirical evaluation shows that this approach is effective at enhancing several exploration mechanisms in a variety of classical planning domains.

1 Introduction

Greedy Best First Search (GBFS) (Doran and Michie 1966) is known to be susceptible to flaws in the heuristic function. However, this can be mitigated by using *stochastic exploration*, with many successful techniques showing improvements in coverage and search time (Valenzano et al. 2014; Xie et al. 2014; Xie, Müller, and Holte 2014; Asai and Fukunaga 2017; Kuroiwa and Beck 2022; Tomasz and Valenzano 2024). These techniques range from uniform sampling to more elaborate bucketing and biased sampling procedures.

The behaviour of GBFS without exploration is characterized by the *Bench Transition System (BTS)* (Heusner, Keller, and Helmert 2018b). A BTS contains the subset of states that GBFS may expand during search, depending on tie-breaking. The BTS is further grouped into connected sets of states called *benches*, and GBFS has made *progress* when it transitions from one bench to another. Given a specific tiebreaking rule, GBFS will traverse a single path of benches through the BTS on its way to a goal.

In this work, we show that one benefit of exploration is its ability to reach easier to solve parts of the state-space. In particular, we analyze several state-spaces and find that in cases where exploration helps, there are often many "easy" bench paths reachable from the initial state, and many of these "easy" states are not on the BTS. That is, these state-spaces have many states that standard GBFS can solve easily, but these states are not found without adding exploration.

Applying this insight, we propose a simple enhancement to exploration called *Locally Greedy Probes*. This technique is motivated by the fact that even if exploration generates a state at the beginning of an easy bench path, it may not be prioritized if it has a high heuristic value. This method therefore briefly pauses the search any time a state is found with a better heuristic value than its parent and performs a hill-climb along a single path in order to make quick progress through easy bench paths. By empirically evaluating locally greedy probes on classical planning problems, we show that this method can improve both coverage and search time of a variety of exploration mechanisms.

2 Background

A *state-space problem* is defined as a set of *states* S, an *initial state* s_I, a set of *goal states* $S_G \subseteq S$, and a *successor function* $succ : S \mapsto 2^S$ where $succ(s)$ are the child states of s. GBFS is a best-first search algorithm commonly used for solving such problems. On each iteration, GBFS selects for expansion the state with the lowest *heuristic value* according to some heuristic function $h : S \mapsto \mathbb{Z}^{\geq 0}$.

Following Heusner, Keller, and Helmert, we briefly review BTS terminology (2017). The *high-water mark* of state s, denoted $hwm(s)$, is the largest heuristic value that **must** be traversed to reach a goal state from s. State s is a progress state if there exists a successor $s' \in succ(s)$ such that $hwm(s) > hwm(s')$. Progress is so defined because upon expanding such a state s, for the remainder of the search, GBFS will only expand a state s' if $hwm(s') < hwm(s)$ (Heusner, Keller, and Helmert 2018b).

For a progress state s, the *bench* of s, denoted $B(s)$, is a set of states defined as follows. Where the *bench level* of s is $level(s) = \min_{s' \in succ(s)} hwm(s')$, for any state $s' \in B(s)$ it holds that $h(s') \leq level(s)$. In addition, s' must be reachable from s along a path containing only non-progress states (aside from s and possibly s'). If s' is a progress state, it is an *exit* of $B(s)$ as it leads to a bench with a lower level.

A *Bench Transition System* is a directed graph whose vertices are benches and whose edges correspond to the exits leading to lower level benches. Each bench is an 'episode' during a GBFS; upon expanding a progress state s, GBFS will remain on the bench of s until an exit state is expanded (Heusner, Keller, and Helmert 2018a). Depending on tie-breaking, GBFS will traverse a single bench path with monotonically decreasing levels. Different bench paths may be easier or harder for GBFS to get through.

GBFS is often enhanced with *exploration* — which occasionally selects states stochastically for expansion. Below, we consider three methods that all alternate between selecting nodes greedily according to h and sampling nodes stochastically. The first is *ϵ-GBFS*, which samples states uniformly randomly from the open list with probability ϵ, where $0 \leq \epsilon \leq 1$ (Valenzano et al. 2014).

Next, we consider *Type-GBFS*, which partitions the open list into different buckets by the tuple $\langle h, g \rangle$ (Xie et al. 2014). This bucketing criteria is called a *type system*. An exploration step in Type-GBFS samples a bucket uniformly at random, then a state therein uniformly at random.

Finally, Softmin-Type extends Type-GBFS by adding biased type selection over $\langle h, g \rangle$ (Kuroiwa and Beck 2022). It first samples an h-value in a way that favours lower h-values, then chooses a random g-cost to complete the tuple. A state is then chosen randomly from the selected bucket.

Asai and Fukunaga (2017) identified that exploration methods can introduce both *inter-plateau* and *intra-plateau* exploration (tie-breaking). Here, a *plateau* refers to states in the open list which share an h-value. Inter-plateau exploration is especially relevant to our study, as it involves varying the order that plateaus are selected from. In contrast, GBFS always selects from the minimum h-value plateau. Selecting from higher plateaus can mean adding states to the open list from multiple bench paths (while GBFS would only traverse one) and so inter-plateau exploration can yield *inter-bench* exploration.

3 Exploration and Bench Paths

In this section, we consider the question of why stochastic exploration can improve search performance. We hypothesize that exploration is most helpful in domains where there are states off the BTS that can be easily solved by GBFS alone. We then analyze individual problems from domains where exploration both hurts and helps to better understand how the presence of such states affects performance.

3.1 The Value of Exploration

We begin by replicating previous studies on using random exploration for solving planning problems. Experiments were run on a VMware Virtual Platform using an 8 core, Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz with a 16 KiB L1 cache, with a 10 minute time limit and a 3.5 GB memory limit per instance. All methods were implemented in Fast Downward (Helmert 2006; Artificial Intelligence Group - University of Basel 2023), using the unit cost FF heuristic (Hoffmann and Nebel 2001). We ran each method 5 times and averaged the results.

Table 1 shows the coverage of ϵ-GBFS with $\epsilon = 0.2$ (column 3), Type-GBFS (column 5), and Softmin-Type (column 7) on 360 problems from 18 domains from the satisficing tracks of IPC 2014 and IPC 2018. Other methods included in the table will be discussed below. As the table shows, all exploration methods outperformed GBFS (column 2) by a large margin. This largely aligns with previous studies (Asai and Fukunaga 2017; Kuroiwa and Beck 2022; Valenzano et al. 2014; Xie et al. 2014; Tomasz and Valenzano 2024).

To understand why exploration helps where it does, we turn to the BTS. Recall that the BTS identifies all bench paths that GBFS may take, and GBFS will go down one depending on tie-breaking. When enhanced with exploration, the search may expand states that are not on the current bench or even on the BTS. In doing so, exploration may allow the search to consider paths to the goal state that GBFS is prevented from reaching by its very nature. With that in mind, we make the following hypothesis:

Hypothesis 3.1. *Domains which contain many states that are not on the BTS but which are easy for GBFS to find a solution from will stand to benefit more from random exploration than domains with few of such states.*

The hypothesis suggests that inter-plateau exploration is beneficial when it allows the search to find states that GBFS alone can solve easily. That is, if we initiated GBFS from these states, it would solve the problem more quickly than staying in the current bench. We hypothesize that state-spaces that have more of such states will benefit more from inter-plateau exploration. This is motivated by the fact that the existing exploration-based approaches all alternate between greedy and exploratory state expansions, or are otherwise biased to prefer states with a low heuristic value. Thus, if the exploration mechanism results in an "easy-for-GBFS" state making it to the top of the open list, then the greedy portion of the search should be able to quickly find a goal.

3.2 State-Space Profiling

To evaluate the hypothesis above, we focus on four domains where exploration had various levels of impact. We used the PDDL-Generators tool (Seipp, Torralba, and Hoffmann 2022) to generate a single problem instance for each domain that was small enough that we could enumerate every state reachable from the initial state. We then calculated the high-water mark of all found states and identified all progress states, even those not on the initial state's BTS. Treating each progress state as an initial state, we ran GBFS with different random tie-breaking 100 times on each. We refer to the average number of expansions to solve each progress state as the "difficulty" of that state's BTS. We limit our view to progress states as our working premise is that inter-plateau exploration helps by landing the search on easier bench paths, and benches are defined by their progress state.

Figure 1 summarizes our findings. On the horizontal axis is the bench level. The vertical axis buckets problems by difficulty. That is, states are in the same bucket if the average number of expansions needed to solve them is in the same order of magnitude. A state is in bucket 0 if it was solved in 0-9 expansions on average, bucket 1 means it was solved

Domain	GBFS	ϵ-GBFS	ϵ_p-GBFS	Δ_p	Type	Type$_p$	Δ_p	Softmin	Softmin$_p$	Δ_p
barman-14	4	7.6	17.6	**+10.0**	9.0	16.8	**+7.8**	15.4	18.8	**+3.4**
childsnack-14	1	0.4	0.0	*-0.4*	0.2	0.0	*-0.2*	0.0	0.2	**+0.2**
floortile-14	2	2.0	3.2	**+1.2**	2.6	2.2	*-0.4*	2.0	2.0	0.0
ged-14	20	20.0	20.0	0.0	17.8	20.0	**+2.2**	20.0	20.0	0.0
hiking-14	20	20.0	20.0	0.0	20.0	20.0	0.0	20.0	20.0	0.0
openstacks-14	0	1.8	1.0	*-0.8*	0.0	0.0	0.0	14.6	15.8	**+1.2**
parking-14	4	4.8	4.8	0.0	1.8	3.0	**+1.2**	8.2	10.2	**+2.0**
tetris-14	6	5.8	6.8	**+1.0**	6.6	6.4	*-0.2*	15.8	18.0	**+2.2**
thoughtful-14	8	10.4	13.4	**+3.0**	14.0	17.4	**+3.4**	9.6	9.8	**+0.2**
transport-14	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
visitall-14	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
agricola-18	10	9.4	10.8	**+1.4**	10.8	11.8	**+1.0**	10.0	10.4	**+0.4**
data-network-18	4	6.2	7.4	**+1.2**	5.6	7.0	**+1.4**	10.0	13.0	**+3.0**
org-synthesis-18	2	3.0	3.0	0.0	3.0	3.0	0.0	3.0	3.0	0.0
org-synth-split-18	9	9.4	9.4	0.0	9.8	10.4	**+0.6**	9.2	9.0	*-0.2*
snake-18	5	5.0	5.0	0.0	5.2	6.4	**+1.2**	5.0	5.0	0.0
spider-18	8	9.4	10.6	**+1.2**	10.0	9.4	*-0.6*	9.4	9.4	0.0
termes-18	12	12.2	14.0	**+1.8**	13.2	12.2	*-1.0*	12.8	14.2	**+1.4**
Total:	115	127.4	147.0	**+19.6**	129.6	146.0	**+16.4**	165.0	178.8	**+13.8**

Table 1: Coverage for IPC 2014 and 2018 domains, with and without probes. Δ_p shows the difference between two variants. Bold values show where probes helped and italics shows where it hurt performance.

in 10-99 expansions, etc. The cells are coloured by the percentage of BTSs *at that level* that have that difficulty, with the number in the cell being the count of the number of states in that bucket. We now consider each of the domains in turn:

Barman (Figure 1a) is an example of a domain where exploration is very helpful. In the selected problem, there are $53,236$ progress states out of $291,627$ total states. The initial state of the problem has a level of 12 and difficulty of $5,508.3$. The figure shows there are easy BTSs at all levels, including those with a higher level than the initial state which are necessarily not on the initial state's BTS.

Visitall (Figure 1b) is a domain in which the FF heuristic provides poor guidance, and exploration also does not help. The selected problem has $158,752$ progress states out of $348,662$ reachable states. The initial state has a level of 21 and difficulty of 64.1. All progress states at level 14 or higher have the same or worse difficulty than the initial state, so there is little to be gained from inter-plateau exploration.

Parking (Figure 1c) is a domain where only Softmin-Type exploration helped. The generated instance has $379,785$ progress states out of $604,800$ total states. The initial state has a level of 16 and difficulty of 100.4. The results are more similar to Barman than Visitall: 85% of BTSs at the initial state's level are easier then the initial state, and very easy benches become available at level 12.

Floortile (Figure 1d) is another example of a domain that is difficult on its own and exploration provides minimal gains. The Floortile instance contained $13,668$ reachable progress states out of $14,607,504$ total states. The initial state has a level of 15 and a difficulty of 134.7. Most BTSs at level 15 or higher are in the initial state's bucket, and its only after stepping down 8 levels that the easiest BTSs are available.

To connect this analysis to Hypothesis 3.1, suppose GBFS is being used to solve a Barman instance, and is currently on a bench at level 11 in the 3rd difficulty bucket. Adding the inter-bench exploration makes the much easier BTSs (buckets 2 and 1) at level 11, 12, or 13 available to the search. There is little opportunity for such a maneuver in Visitall and Floortile. In these domains, inter-bench exploration is most likely to land the search on a BTS of the same difficulty or harder. Thus, we would expect inter-plateau exploration to be more helpful in domains like Barman or Parking, where these alternative easy BTSs are available.

4 Locally Greedy Probes

We now introduce *Locally Greedy Probes*, which aim to further enhance the ability of exploration techniques to take advantage of "easy" progress states when they encounter them. To motivate probes, consider the Barman problem in Section 3.2. If ϵ-GBFS expands an "easy" progress state at a higher level than the initial state, then its successor will remain near the back of the open list and thus ignored by the greedy selections, and will have a low probability of being selected stochastically if the open list is already large.

Locally greedy probes can be added to any method incorporating random exploration to help in such cases. When doing so, a probe is initiated whenever the search expands a state p that has a successor s with a lower h-value than p. When this happens, the main search is paused and a probe is started from s. A probe consists of a single path, generated by performing a local hill-climb as shown in Algorithm 1. The probe will continue as long as the current state s has a child that makes heuristic improvement (line 6) that avoids the open and closed lists. The next state added to the probe will be one of the most improving successors (line 7), while the remaining successors will be added to OPEN (line 10). The probe continues until either a goal is reached or no further local heuristic improvement is possible (line 9).

Continuing the Barman example, a probe would immediately expand the most improving successor s of that higher level progress state. If that s is also a progress state, the probe will continue down into the easy bench path. If not, the

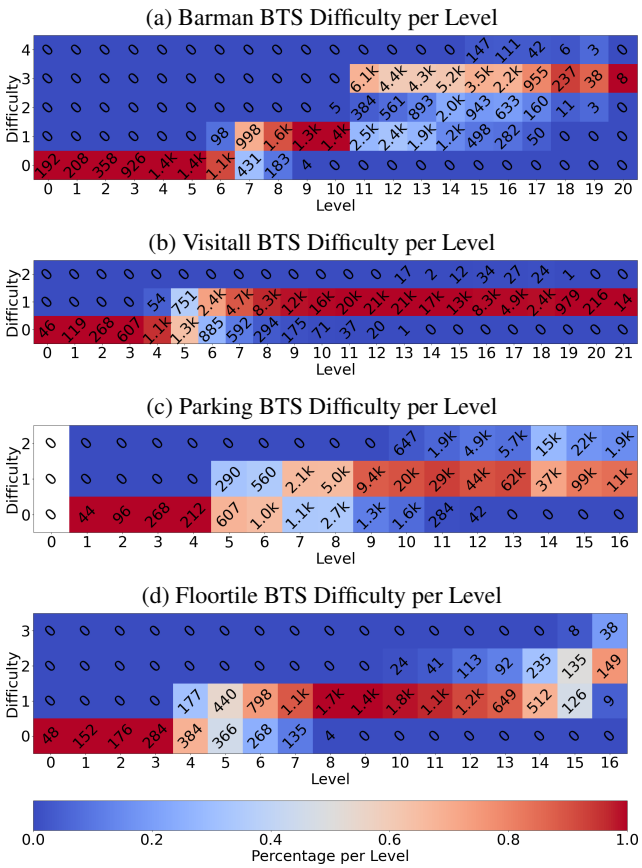

Figure 1: Difficulty profiles.

probe will only perform one additional expansion. Probes therefore allow the search to make quick progress along easy bench paths that exploration alone may take time to return to, and they do so without much overhead. Importantly, a probe is initiated whenever a progress state is encountered since every progress state s has an improving successor.

Table 1 shows the per-domain performance when adding probes to ϵ-GBFS (column 4), Type-GBFS (column 6), and Softmin-Type (column 8). The variants which use probes — denoted by a subscript p — saw a large improvement in coverage. This is captured by the Δ_p column, which shows the per-domain difference of using probes for each method. Notably, probes never hurt any method by more than a single

Algorithm 1: Locally Greedy Probe

Require: a state s, open list OPEN, closed list CLOSED
while $s \neq$ null **do**
  if s is a goal **then return** found solution path to s
  $S \leftarrow \{s' \in succ(s) \mid s' \notin$ OPEN $\cup$ CLOSED$\}$
  add s to CLOSED
  $h_B \leftarrow \min_{s' \in S} h(s')$
  if $h_B < h(s)$ **then**
    $s \leftarrow$ random node from $\{s' \in S \mid h(s') = h_B\}$
  else
    $s \leftarrow$ null
  OPEN $\leftarrow$ OPEN $\cup (S \setminus s)$
return null

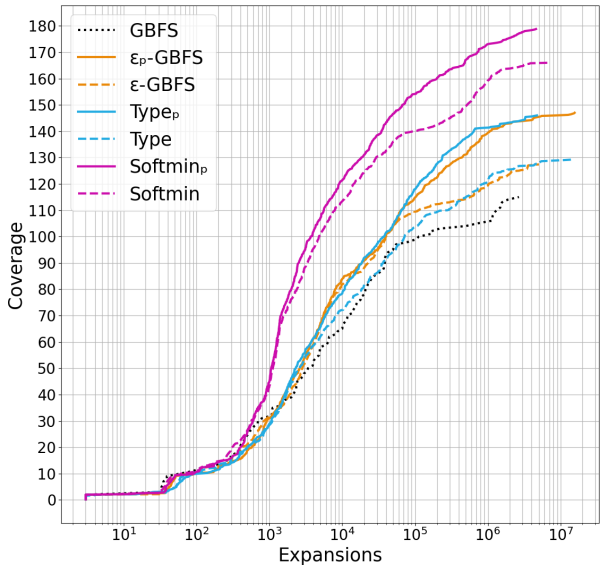

Figure 2: Expansions vs Coverage for all 5 runs.

problem on any domain. Another view of this data is shown in Figure 2, which plots the average coverage against the average number of expansions. The figure shows that after the easy problems are solved, the use of probes dominates not having probes for all numbers of expansions.

Recall that probes are explicitly designed to take advantage of domains with more states with easier BTSs. Thus, given the analysis in Section 3.2, we would expect probes to have more benefit in those domains like Barman and Parking, than in those domains like Visitall and Floortile. Indeed this is what we see: probes markedly improved performance in Barman, marginally improved it in Parking, the impact was mixed in Floortile, and was non-existent in Visitall.

Lastly, we would expect probes to show up more often in the found plans in domains like Barman than in domains like Visitall. To test this, we ran ϵ_p-GBFS 5 times on each of the 34 Barman and 40 Visitall problems from the optimal tracks of IPC 2011 and 2014. We chose the optimal tracks so that all instances would be solved. Probes showed up in 99.4% Barman solution paths and 70.5% of Visitall solution paths, which is consistent with the intuition given above.

5 Conclusion and Future Work

In this paper, we provided evidence that exploration-based GBFS enhancements improve performance when there are "easy" states in the state-space that GBFS alone cannot find. We then show that a simple approach called locally greedy probes can help exploration take advantage of such states, leading to improved coverage and speed.

For future work, the analysis could be extended to multi-heuristic best-first search (Helmert 2006), novelty (Lipovetzky and Geffner 2017), or other planner enhancements, to see if they also help the search find easy progress states. Testing probes with other type systems like those that approximate the BTS also remains an interesting possibility (Tomasz and Valenzano 2024).

Acknowledgements

We thank the anonymous reviewers for their helpful feedback on this work. We also gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Artificial Intelligence Group - University of Basel. 2023. Fast Downward. https://github.com/aibasel/downward.

Asai, M.; and Fukunaga, A. 2017. Exploration among and within plateaus in greedy best-first search. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27, 11–19.

Doran, J. E.; and Michie, D. 1966. Experiments with the Graph Traverser Program. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 294(1437): 235–259.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Heusner, M.; Keller, T.; and Helmert, M. 2017. Understanding the search behaviour of greedy best-first search. In *Proceedings of the International Symposium on Combinatorial Search*, volume 8, 47–55.

Heusner, M.; Keller, T.; and Helmert, M. 2018a. Best-case and worst-case behavior of greedy best-first search. International Joint Conferences on Artificial Intelligence.

Heusner, M.; Keller, T.; and Helmert, M. 2018b. Search progress and potentially expanded states in greedy best-first search. International Joint Conferences on Artificial Intelligence.

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

Kuroiwa, R.; and Beck, J. C. 2022. Biased Exploration for Satisficing Heuristic Search. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, 213–221.

Lipovetzky, N.; and Geffner, H. 2017. Best-first width search: Exploration and exploitation in classical planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Seipp, J.; Torralba, Á.; and Hoffmann, J. 2022. PDDL Generators. https://doi.org/10.5281/zenodo.6382173.

Tomasz, D.; and Valenzano, R. 2024. The Bench Transition System and Stochastic Exploration. In *Proceedings of the International Symposium on Combinatorial Search*, volume 17, 143–151.

Valenzano, R.; Sturtevant, N.; Schaeffer, J.; and Xie, F. 2014. A comparison of knowledge-based GBFS enhancements and knowledge-free exploration. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 24, 375–379.

Xie, F.; Müller, M.; and Holte, R. 2014. Adding local exploration to greedy best-first search in satisficing planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Xie, F.; Müller, M.; Holte, R.; and Imai, T. 2014. Type-based exploration with multiple search queues for satisficing planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.