Exploration Among and Within Plateaus in Greedy Best-First Search

Submision 229

Abstract

Recent enhancements to greedy best-first search (GBFS) such as DBFS, ϵ-GBFS, Type-GBFS improve performance by occasionally adopting a non-greedy node expansion policy, resulting in more exploratory behavior. However, previous exploratory mechanisms do not address exploration within the space sharing the same heuristic estimate (plateau). In this paper, we show these two modes of exploration, which work across (*inter-*) and within (*intra-*) plateau, are complementary, and can be combined to yield superior performance. We also introduce IP-diversification, a method combining Minimum Spanning Tree and randomization, which addresses "breadth"-bias instead of the "depth"-bias addressed by the existing diversification methods. We evaluate IP-diversification for both intra- and inter-plateau exploration, and show that it significantly improves performance in several domains. Finally, we show that combining diversification methods results in a planner which is competitive to the state-of-the-art for satisficing planning.

1 Introduction

Many search problems in AI are too difficult to solve optimally, and finding even one satisficing solution is challenging. Greedy Best-First Search (GBFS) is a best-first search variant where $f(n)$, the expansion priority of node n is based only on a heuristic estimate of the node, i.e., $f(n) = h(n)$ (in contrast with A^* search, where the node priority also considers $g(n)$, the cost from the start state to n, and $f(n) = g(n) + h(h)$). Although GBFS ignores solution optimality, it has been shown to be quite useful when it is necessary to find some satisficing solution quickly, and GBFS has been the basis for state-of-the-art domain-independent planners.

Despite the ubiquitous use of GBFS for satisficing search, previous work has shown that GBFS is susceptible to being easily trapped by undetected dead ends and huge search plateaus. On infinite graphs, GBFS is not even complete (Valenzano and Xie 2016) because it could be misdirected by the heuristic guidance forever. These pathological behaviors are caused by the fact that the search behavior of GBFS strongly depends on the quality of the heuristic function.

The problem is exacerbated by the fact that GBFS tends to be combined with inadmissible heuristic functions such as the FF heuristic (Hoffmann and Nebel 2001b), Causal Graph (Helmert 2006) or Landmark-count (Richter, Helmert, and Westphal 2008) heuristics. An inadmissible heuristic can cause nodes which are close to the goal (low h^*, optimal cost to goal) to be incorrectly labeled as unpromising (overestimation: $h > h^*$), causing GBFS to delay expanding them until all other nodes in the current local minima with smaller h-values have been expanded.

Recently, several approaches have been proposed for alleviating this problem, e.g., DBFS (Imai and Kishimoto 2011), ϵ-GBFS (Valenzano et al. 2014) and Type-GBFS (Xie et al. 2014). They improve the search performance by occasionally expanding nodes which do not have the lowest h-value, i.e., diversifying the search. These diversified algorithms provides an opportunity to expand nodes that are mistakenly overlooked due to errors made by the heuristic functions. A common objective among these methods is the removal of some (undesirable/unintended) *bias*, thereby encouraging *exploration* by the search process and adding *diversity* in decision making process. In this paper, we use the terms "exploration", "diversity", and "bias removal" interchangeably. Existing methods for exploration have two issues: First, previous methods all employ h-based diversification as part of their algorithms in order to avoid the bias toward the nodes with smaller estimates. However, h-based diversification cannot detect the bias *among nodes with the same h-cost*. Second, as we see later, they are based on diversification with respect to search depth (distance from the start / goal / plateau entrance), so the bias among the set of nodes with the same search depth is not removed.

We first show that a recently proposed depth-based tie-breaking strategy for A^* (Asai and Fukunaga 2016) also improves the performance of GBFS by diversifying the depth within each h-plateau. Both depth diversification strategy and Type-GBFS are instances of a *type*-based diversification strategy (Xie et al. 2014): Depth diversification applies type-based diversification within a plateau, and Type-GBFS applies it between plateaus. We compare their empirical performance and show that their improvements are complementary – Two configurations improve the performance in different domains, and a configuration using both methods benefits from both, achieving the better coverage. This effectively shows that *inter*-plateau and *intra*-plateau diversification are two orthogonal usages of diversification.

Next, we propose and evaluate a new diversification strategy called IP-diversification which addresses diversity with respect to *breadth*. We evaluate this new diversification strategy both for intra-plateau and inter-plateau exploration. Complementary effects on intra/inter-plateau exploration were observed. In addition, IP-diversification outperforms the Type-based diversification strategy. Finally, we show that by combining several intra/inter plateau exploration strategies, we can improve upon state-of-the-art planners in terms of coverage.

2 Preliminaries and Background

We first define some notation and the terminology used throughout the rest of the paper. $h(n)$ denotes the estimate of the cost from the current node n to the nearest goal node. $g(n)$ is the current shortest path cost from the initial node to the current node. $f(n) = g(n) + h(n)$ is the estimate of the resulting cost of the path to a goal containing the current node. We omit the argument (n) unless necessary. h^*, g^* and f^* denotes the true optimal cost from n to a goal, from the start to n, or from the start to a goal through n, respectively.

A *sorting strategy* for a best first search algorithm tries to select a single node from the open list (OPEN). Each sorting strategy is denoted as a vector of several *sorting criteria*, such as [criterion$_1$, criterion$_2$, ..., criterion$_k$], which means: First, select a set of nodes from OPEN using criterion$_1$. If there are still multiple nodes remaining in the set, then break ties using criterion$_2$ and so on, until a single node is selected. The *first-level sorting criterion* of a strategy is criterion$_1$, the *second-level sorting criterion* is criterion$_2$, and so on.

Using this notation, A^* without any tie-breaking can be denoted as $[f]$, and A^* which breaks ties according to h value is denoted as $[f, h]$. Similarly, GBFS is denoted as $[h]$. Unless stated otherwise, we assume the nodes are sorted in increasing order of the key value, and BFS always selects a node with the smallest key value.

A sorting strategy fails to select a single node when some nodes share the same sorting keys. In such cases, a search algorithm must select a node according to a *default* tie-breaking criterion, criterion$_k$, such as *fifo* (first-in-first-out), *lifo* (last-in-first-out) or *ro* (random ordering). For example, an A^* using h and *fifo* tie-breaking is denoted as $[f, h, \mathit{fifo}]$. By definition, default criteria are guaranteed to return a single node from a set of nodes. When the default criterion does not matter, we may use a wildcard $*$ as in $[f, h, *]$.

Given a search algorithm with a sorting strategy, a *plateau* (criterion ...) is a set of nodes in OPEN whose elements share the same sort keys according to non-default sorting criteria and therefore are indistinguishable. In a case of A^* using tie-breaking with h (sorting strategy $[f, h, *]$), the plateaus are denoted as *plateau* (f, h), the set of nodes with the same f cost and the same h cost. We can also refer to a specific plateau with $f = f_p$ and $h = h_p$ by *plateau* (f_p, h_p).

Finally, OPEN list *alternation* (Röger and Helmert 2010) is a technique to combine multiple sorting strategies in order to improve the robustness of the search algorithm. Nodes are simultaneously stored and sorted into independent OPEN lists with different strategies, and node expansion alternates among the OPEN lists. We denote an alternating OPEN list as $\mathit{alt}(X_1, X_2, \ldots)$ where each X_i is a sorting strategy.

Depth-Based Tie-breaking To date, there has been relatively little work on tie-breaking policies for BFS. Recently, Asai and Fukunaga (2016) performed an in-depth investigation of tie-breaking strategies for A^*, in which the tie-breaking policy was found to have a significant effect on the performance when the search plateau is huge. In the most commonly used sorting strategies, $[f, h, \mathit{fifo}]$ or $[f, h, \mathit{lifo}]$, the search has a strong bias to focus on either the regions of smaller (*fifo*) or larger (*lifo*) search depth of each *plateau* (f, h), causing failure to find the solution within a given time limit.

To address the issue caused by the search bias within a plateau, they proposed a notion of *depth* and diversified the search over different depths within a plateau. The depth $d(n)$ of a state n is an integer representing the step-wise distance from the *entrance* of the plateau (the most recent state which entered the plateau, along the path from the initial state). $d(n) = d(m) + 1$ when n and the parent node m are on the same plateau. For example, with strategy $[f, h, *]$, *plateau* $(f(n), h(n))$ = *plateau* $(f(m), h(m))$, therefore $f(n) = f(m) \wedge h(n) = h(m)$. $d(n)$ is 0 otherwise. The nodes are stored in buckets indexed by depth, and expansions are allocated across different buckets with equal probability at every iteration. The resulting sorting strategy is denoted as $[f, h, \langle d \rangle, *]$.

For GBFS, to our knowledge, there is currently no well-established tie-breaking policy analogous to h-based tie-breaking for A^*. Presumably, this is because while A^* has access to three cost values (f, g, and h), GBFS is guided solely by the heuristic value h.[1] As a consequence, improvements to GBFS have been primarily achieved by addressing other aspects, such as modifying the evaluation scheme (Richter and Westphal 2010, lazy evaluation), queue alternation (multiple heuristic functions), preferred operators (Hoffmann and Nebel 2001b), and diversification.

Exploration Mechanisms One recent class of improvements to GBFS seeks to introduce exploration (diversity) to the search process, as exemplified by DBFS (Imai and Kishimoto 2011), ϵ-GBFS (Valenzano et al. 2014), Type-GBFS (Xie et al. 2014). These algorithms address the problem of GBFS getting stuck due to heuristic errors. In GBFS, a node will not be expanded until it expands all nodes with a lower h-value in the current local minima. Thus, search progress can be delayed when a good (low-h^*) node is mistakenly assigned a poor (high) h-value (overestimation), or bad (high-h^*) nodes are assigned promising h-values (low-h, underestimation). These exploration strategies allow the search to escape the local minima by relaxing the h-based best-first node expansion order.

KBFS(k) (Felner, Kraus, and Korf 2003) attempts to address this problem by expanding k nodes at a time. ϵ-GBFS

[1] Tie-breaking based on g is sometimes used, but this is motivated as a means to find higher-quality solutions. To our knowledge, in a satisficing context, tie-breaking strategies for reducing search effort have not been explicitly motivated or evaluated.

(Valenzano et al. 2014) selects a random node from OPEN with some fixed probability $\epsilon < 1$. This is a randomized, weighted alternating OPEN list using $[h,*]$ and $[ro]$ (no sorting criteria): $alt([h,*],[ro])$.

While ϵ-GBFS relies on a pure randomization strategy to escape traps and introduce exploration, Type-GBFS (Xie et al. 2014) explicitly seeks to remove bias and diversify the search by categorizing OPEN according to several key values, such as $[g,h]$ for each state. Each node is assigned to a bucket according to its key value. The search then selects a random node in a random bucket, avoiding the cardinality bias among buckets. Since Type-GBFS does not sort the buckets according to the key vector, we use a different notation $\langle \ldots \rangle$, such as $\langle g,h \rangle$ denoting type buckets whose key values are g and h. In the implementation evaluated by Xie et al. (2014), Type-GBFS alternates the exploitative (standard best-first order) expansion and the exploratory (randomized) expansion. We denote this as $alt([h,*],[\langle g,h \rangle, ro])$.

DBFS (Imai and Kishimoto 2011) diversifies the search based on g and h values, but with several key differences from the above two algorithms: First, the exploratory selection is not uniformly random, but is subject to a particular distribution function based on h, g, h_{min} and g_{max}. Second, it uses a local search with a bounded number of expansions equal to $h(s)$, which dynamically balances the exploration and exploitation — it does more GBFS when h is large (far from the goal), and less GBFS near the goal (h is small).

GBFS with Local Exploration (GBFS-LE), introduces a 2-level search architecture which runs GBFS until it detects that no improvements have been made for a while, and then runs a local search (GBFS-LS) or random walk (GBFS-LRW) in order to find an exit to a more promising region of the search space (Xie, Müller, and Holte 2014).

3 Intra- and Inter-plateau Diversification

Previous work on exploration for GBFS address the problem of heuristic errors by occasionally expanding nodes with high h, both in order to avoid expanding more dead-end states (high h^*) labeled as promising (low h), or to find a good state (low h^*) labeled as unpromising (high h). Since this type of diversification operates across different search plateaus, we refer to these as *inter-plateau* exploration. However, we propose another type of exploration, which we call *intra-plateau* exploration, which works within a particular plateau: This type of exploration only changes expansion order among the nodes within a plateau.

Existing inter-plateau exploration can be understood as a diversification applied to h^* plateau. Consider a hypothetical 2-dimensional histogram (Figure 1) of the number of nodes for each pair h, h^*. If both axes were h^* (i.e., h is a perfect heuristic), all nodes would be on the diagonal line $x = y$. However, in reality, h has errors relative to h^*, as would be shown if we projected the histogram to the x-axis. Since low-h^* nodes may have high-h values, it is sometimes reasonable to expand high-h nodes depending on the distribution defined by the problem characteristics and the heuristic function. To our knowledge, all previous work on exploration for GBFS has addressed exploration along this dimension by ignoring the best-first ordering wrto h.

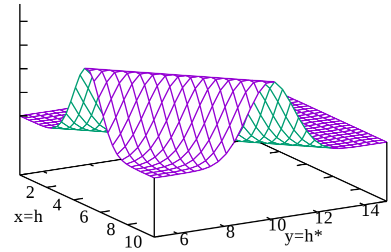

Figure 1: A conceptual view of the node distribution wrto h^* and inadmissible h. The peak line on the surface is on $x = y$. Projection to x-axis shows the distribution of h values, while projection to y-axis shows the distribution of h^* values.

However, not only can a single h^*-plateau consists of nodes with different h values, the converse can also be true – a single h-plateau consists of nodes with different h^* values, as would be shown by projecting the histogram to the y-axis in Figure 1. This leads to an observation that *in the worst case, a naive algorithm may keep expanding bad (high-h^*) nodes within an h-value plateau*. This pathological behavior happens for each h-plateau and is not explicitly addressed by the previous diversification techniques because they do not modify intra-plateau behavior.

Unlike inter-plateau exploration, which addresses ***incorrect information for h*-plateau****, intra-plateau exploration addresses the problem of* ***insufficient information for h-plateau.*** An intra-plateau exploration strategy tries to avoid the aforementioned pathology of continually expanding high-h^* nodes within an h-value plateau. Since we do not know *a priori* which nodes in an h-plateau are better (low-h^*) than the other nodes in the same h-plateau, by an adversary argument, the safest strategy is to avoid biased choices – in the absences of useful heuristic knowledge which differentiates among a set of nodes, an expansion policy which is biased to expand some particular group of states within a plateau can be exploited by an adversary which seeks to hide better (low-h^*) nodes.

Type-Based Diversification The notions of inter-vs-intra plateau exploration allows us to discuss and compare depth diversification (Asai and Fukunaga 2016) and Type-GBFS (Xie et al. 2014) within a unified framework – it turns out that these are essentially the same algorithm, except that they are using different key values (metrics) in different contexts (inter-vs-intra plateau, satisficing-vs-optimal search).

Lelis, Zilles, and Holte (2013) define a general framework for adding exploration to search using "type systems":

Definition 1. *A* Type system *(Lelis, Zilles, and Holte 2013) is a function from a node to a vector, T : node $\rightarrow \mathbb{Z}^k, T(n) = \langle t_1(n) \ldots t_k(n) \rangle$, where each function $t_i(n)$ returns an integer for each node n.*

Xie et al. proposed a node selection technique based on type systems.

Definition 2. Type-Based Node Selection *(Xie et al. 2014) with a type system $T(\cdot)$ of k types maintains a k-dimensional matrix of sets of nodes, where each set S_v is associated with a vector $v = \langle v_1, \ldots, v_k \rangle$. Each node n is stored in $S_{T(n)}$.*

For dequeueing, it randomly selects a non-empty set from all sets, and a random node in the set is dequeued.

The reason for selecting a set at random is to try to allocate the search effort among a diverse set of nodes. Some sets could contain a large number of nodes while others are only scarcely populated. Type-based node selection tries to remove this cardinality bias among buckets. Because type-based node selection has this diversification as an explicit goal and is best understood as a diversification strategy, we call it ***type-based diversification*** in the rest of this paper.

Type-GBFS (Xie et al. 2014) uses type-based diversification with type system $\langle g, h \rangle$ for inter-plateau exploration. Their inter-plateau exploration is implemented by queue alternation (Röger and Helmert 2010) between standard Best-First queue and type-based diversification queue.

Depth diversification (Asai and Fukunaga 2016) originally addressed the issue of zero-cost actions in admissible search with A^*, and the configuration was denoted as $[f, h, \langle d \rangle]$. In order to use $\langle d \rangle$ for GBFS, the resulting configuration is $[h, \langle d \rangle]$, and the depth d is defined for each h-plateau. This configuration is considered as an instance of intra-plateau type-based diversification because it uses type-based diversification with type system $\langle d \rangle$ for diversifying the search within plateaus defined by h.

3.1 Empirical Comparison of Intra- and Inter-Plateau Exploration

Since depth-diversification and Type-GBFS turned out to be instances of the same strategy applied for different purposes (intra/inter-plateau), we use these as exemplars to compare the impact of intra/inter-plateau exploration. In the following experiments, we empirically show that they achieve complementary performance improvements. This indicates that inter/intra-plateau exploration in fact addresses orthogonal issues of *incorrect* and *insufficient* information, respectively. We then show that intra/inter-plateau exploration can be successfully combined in a single search algorithm.

We compare the performance of the following configurations for Greedy best-first search using the Fast Forward heuristic h^{FF} (Hoffmann and Nebel 2001a) and Causal Graph heuristic h^{CG} (Helmert 2004).

- **h**: baseline GBFS (eager evaluation).
- **hd**: Depth diversification (Asai and Fukunaga 2016) – intra-plateau type-based diversification, $[h, \langle d \rangle]$.
- **hD**: Type-GBFS (Xie et al. 2014) – inter-plateau type-based diversification, $alt([h], [\langle g, h \rangle, ro])$,
- **hdD**: A combined configuration of intra- and inter-plateau type-based diversification, $alt([h, \langle d \rangle], [\langle g, h \rangle, ro])$.

Experiments are conducted on a Xeon E5-2666 @ 2.9GHz, HyperThreading and TurboBoost disabled. We used a 4GB memory limit and 5 minutes time limit, on IPC 2011 and 2014 instances. Since IPC 2011 and IPC 2014 contain duplicate domains, we removed duplicates from the 2011 set, keeping the 2014 versions. All implementations are based on FastDownward (Helmert 2006) and unless specified, all configurations use *fifo* default tiebreaking (FastDownward default).

Following previous work (Valenzano et al. 2014; Xie et al. 2014), all configurations are evaluated under unit cost transformation because in these experiments, we focused on the coverage (number of problems solved within resource limit) for purely satisficing search. Each experiment is run 10 times, and the means are shown in Table 1.

First, intra-plateau exploration **hd** increases coverage for both heuristics h^{CG} (187 $\rightarrow$ 194.2) and h^{FF} (192 $\rightarrow$ 223.9). This shows that intra-plateau exploration successfully allows GBFS to avoid being trapped in h-value plateaus. Inter-plateau exploration **hD** also increases coverage for both heuristics, confirming the results in (Xie et al. 2014). It is worth mentioning that the performance of **hd** is comparable to **hD**, showing that intra-plateau exploration is no less important than inter-plateau exploration which previous work focused on.

Second, the data shows that the effects of inter/intra-plateau exploration are complementary, as would be expected since they are designed to address orthogonal issues. In most cases, when **hd** improves upon **h** then **hdD** improves upon **hD**, and when **hD** improves upon **h** then **hdD** improves upon **hd**. As a result, for both h^{CG} and h^{FF} heuristics, the **hdD** configuration had higher coverage (h^{CG}:215.8, h^{FF}:223.9) than the hd (h^{CG}:194.2, h^{FF}:208) and hD (h^{CG}:206.1, h^{FF}:207.4) configurations. This shows that combining intra/inter-plateau exploration methods which address orthogonal issues results in better overall performance than either type of exploration by themselves.

Based on these results, we conclude that:

1. Inter- and intra-plateau exploration address orthogonal issues and have complementary performance;
2. Combining inter- and intra-plateau exploration can result in better performance than either exploration alone.

4 Breadth-Based Diversification: Invasion Percolation

A limitation of type-based diversification based on path distance is that it does not diversify with respect to breadth – nodes with equal estimated distance from goals (h), initial states (g) or plateau entrance (d) are put in a single set. This makes it susceptible to pathological behavior on graphs where some nodes have many more children than others.

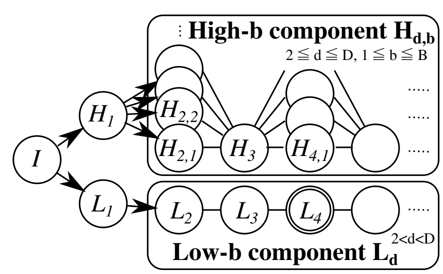

Figure 2: Example case exhibiting large bias in the branching factor depending on the subgraph.

Consider a blind search on the directed acyclic graph shown in Figure 2. The graph consists of two large components, **high-b** and **low-b** branches, and their entries H_1, L_1.

		h^{CG}				h^{FF}			
		h	hd intra	hD inter	hdD both	h	hd intra	hD inter	hdD both
	total	187	194.2	206.1	**215.8**	192	208	207.4	**223.9**
IPC11 w/o duplicates	elevators	9	8	8.7	**9.7**	19	14	15.9	13.7
	nomystery	7	6	**15.4**	**15.1**	9	7	**16.6**	**17**
	parcprinter	20	20	19.4	18.7	20	20	20	20
	pegsol	20	20	20	20	20	20	20	20
	scanalyzer	20	20	19.9	20	15	15.1	**18**	**18.6**
	sokoban	16	16	**16.9**	**17**	19	19	17.4	17.4
	tidybot	16	**18**	**18.7**	**18.6**	16	16	16	**16.7**
	woodwork	2	2	**2.7**	**7.7**	2	2	**4**	**7.2**
IPC14	barman	0	0	0	0	0	0	**1.5**	**1**
	cavediving	7	7	7	7	7	7	7	7.2
	childsnack	1	**6**	0.1	**1.5**	0	**4**	0	0.3
	citycar	0	0	**7.8**	**4.7**	0	0	**7.2**	**7.1**
	floortile	0	0	**2**	**2**	2	2	2	2.1
	ged	0	0	**9.6**	**9.7**	19	19	14	13.8
	hiking	18	16.9	**19.5**	**19.7**	20	20	19.8	20
	maintenance	16	16	16.1	15.8	11	8	10.7	11.1
	openstacks	0	**3.5**	0	**0.5**	0	**12.6**	0	**7**
	parking	7	**9.7**	1.2	4.1	4	**7.5**	1.4	**5.7**
	tetris	18	17.1	12.4	14.3	1	**5.8**	3.2	**4.9**
	thoughtful	5	5	5	5	8	**9**	**12.7**	**13.1**
	transport	5	3	3.7	4.7	0	0	0	0
	visitall	0	0	0	0	0	0	0	0

Table 1: Number of solved instances (5 min, 4GB RAM), mean of 10 runs. **h**: baseline GBFS. **hd/hD**: intra/inter-plateau type-based diversification $[h, \langle d \rangle]$ and $alt([h], [\langle g, h \rangle, ro])$ (Type-GBFS), **hdD**: A combined configuration, $alt([h, \langle d \rangle], [\langle g, h \rangle, ro])$. **Bold** indicates that (improvements vs. baseline)> 0.5. Blue indicates that **hdD** improvement correlates with **hd** (intra-plateau) improvement, red indicates that **hdD** improvement correlates with **hD** (inter-plateau) improvement, and orange indicate that both intra/inter-plateau schemes as well as the combined **hdD** scheme improved. Thus, intra- vs. inter-plateau scheme have complementary effects that improve **hdD**.

The initial search node is I and the goal node is L_4. Both branches have maximum depth D, and the high-b branch has maximum width B. Both B and D are very large. This graph presents a pathological case for all of the previously described methods (*lifo*, *fifo*, *ro* and type-based diversification), depending on successor ordering. *lifo* performs a DFS, and if *lifo* first searches H_1 and the high-b branch due to successor ordering, it must explore the entire high-b branch before expanding L_1 and low-b branch. *fifo* performs Breadth-First Search (BreadthFS), and will therefore suffer from the high branching factor at depth 2 of the high-b branch, getting stuck before reaching L_4. Although randomization can allow *ro* to be better off than the behavior of *fifo*/BreadthFS, but the effect is limited: For example, while expanding depth 2, *ro* may occasionally expand depth 3 because it uniformly randomly selects a node from OPEN. However, the probability of expanding nodes at depth 3 is initially only $1/(B+1)$ and continues to be small until most of the nodes at depth 2 are expanded, because OPEN is mostly populated with the nodes from depth 2. Depth-based diversification addresses the depth bias of BreadthFS. However, even though it distributes the effort among various depths, the probability of expanding L_2, L_4 at depths 2 and 4, is only $1/(B+1)$ each, which is very low when B is very large.

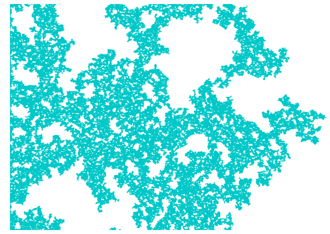

Figure 3: Invasion Percolation on 2-dimensional lattice

We propose *Invasion Percolation-based diversification (IP-diversification)*, a new diversification strategy for satisficing search that addresses this type of bias. IP-diversification combines randomization and Prim's method (Prim 1957) for Minimum Spanning Tree (MST).

Invasion Percolation Invasion Percolation (Wilkinson and Willemsen 1983) simulates the distribution of fluid slowly invading porous media, e.g., water replacing the air in a porous rock. We focus on a variant called bond IP (BIP), where "bonds" indicate edges in a lattice, and present the graph-based description by Barabási (1996). Given initial node(s) and a graph whose edges are assigned independent random values, BIP iteratively marks the nodes. Once assigned, the random value on each edge never changes. The initial nodes are marked by default. In each iteration marks an unmarked node to which the least-value outgoing edge leads. Marked nodes represent the porous sites whose air is replaced by the water (invader). Barabási (1996) showed that this algorithm is equivalent to applying Prim's method for MST (Prim 1957) on a randomly weighted graph: Prim's method constructs an MST by iteratively adding a neighboring edge with the least edge costs to the existing tree.

Figure 3 illustrates a 2-D lattice after running BIP for a while. The initial nodes are at the leftmost edge of the rectangular region, i.e. the fluid percolates from the left. The resulting structure has holes of various sizes that the fluid has not invaded, due to the high-valued edges surrounding the neighbors of the holes, which serve as an embankment preventing the water from invading. Since the random value on each edge is fixed, the algorithm does not mark the nodes inside the hole until it marks all nodes with smaller random values in the entire space outside the embankments. This behavior is critical to forming a fractal structure.

Invasion Percolation for Search Diversification We adapt the BIP model as a exploration mechanism for best-first search. Previous work on BIP was on physical simulations with relatively small graphs, and to our knowledge, this is the first application of BIP to complex *implicit* graphs.

Consider applying BIP to the DAG in Figure 2. There is a non-negligible probability that the search finds the solution without expanding high-b branch: This occurs when the value $v(H_1)$ of H_1 is higher than the value of any of $L_1 \ldots L_4$, whose probability is 1/5 (follows from $\int_0^1 dv(H_1) Pr(\forall i; v(L_i) \leq v(H_1)) = \int_0^1 x^4 dx$). In this case, node H_1 is acting as an embankment, causing nodes in the low-b branch to be expanded. In contrast, the opposite case is very unlikely: L_1 *could* be expanded after expanding all of $H_{d,b}$ for $1 \leq d \leq 4$ and $1 \leq b \leq B$, but the probability of this, $1/(2B+3)$, is very small (assuming large B).

Also consider the case when H_1 is expanded with probability $4/5$. Even if this embankment is broken, H_3 could act as another embankment again with probability $1/5$. Moreover, it also avoids expanding large number of nodes in $H_{2,i}$ whose values are higher than $L_1 \dots L_4$. $B/5$ of the nodes are not expanded on average because each node is not expanded with the same probability $1/5$.

Thus, at every possible "bottleneck" in the search space that forms an embankment, BIP tends to start looking at the other branches. Since this is affected by the least width of a subgraph rather than the maximum, it is less likely to suffer from the pathological behavior exemplified by Figure 2.

The actual implementation of BIP is quite simple: A function r_{BIP} returns a randomly selected value for each search edge that caused the node to be evaluated. For each edge, the function should always return the same value once a random value is assigned to that edge. This requires storage whose size is linear in the number of edges that are explored.

For intra-plateau exploration, r_{BIP} is used to break ties in a plateau induced by the primary heuristic function h, i.e. $[h, r_{\text{BIP}}, *]$. Since nodes are sorted in increasing order of the memoized random value attached to each edge, the node expansion order within a plateau follows that of Prim's method. For inter-plateau exploration, we alternate the expansion between standard GBFS and a queue sorted by r_{BIP}: *alt*$([h], [r_{\text{BIP}}])$, just as in Type-GBFS.

Node expansion order according to r_{BIP} differs significantly from that of *ro* (pure random selection). *ro* is equivalent to performing a random sort and select the first node, i.e., *ro* essentially assigns a *new* random value to *all* nodes at every single expansion. In contrast, r_{BIP} assigns a value to each edge only once, which develops embankments and allows unexplored "holes" to have longer lifetimes. Consider what would happen if we switch the behavior from r_{BIP} to *ro* starting from the state shown in Figure 3. Since all nodes are assigned a new random value at each expansion, the embankment nodes are more likely to be expanded, filling the holes more quickly. Thus, running *ro* results in a more solid, denser expansion biased to the left, near the initial nodes.

There is one difference between the assumptions made by BIP/Prim (Barabási 1996) and classical planning. The search spaces of classical planning are directed while BIP/Prim assumes undirected graphs. Thus, although Prim's method finds the minimum spanning tree on an undirected graph, it may not return the minimum-weight tree on a directed graph. This, however, does not affect the completeness of our search algorithm because it just changes the order of expansion (BIP-based search diversification does not prune any nodes). Adopting algorithms for minimum spanning arborescence for directed graphs (Chu and Liu 1965; Edmonds 1967; Tarjan 1977; Gabow et al. 1986) to search diversification is a direction for future work.

Search Behavior of IP-diversification We analyze the basic search behavior of IP-diversification by applying a blind search on IPC satisficing instances. We ran four configurations, namely Type-based diversification with depth d (hd:$[\langle d \rangle]$) and IP-diversification (hb:$[r_{\text{BIP}}]$), as well as BreadthFS (h:[*fifo*]) and random search (ro:[*ro*]). All solvers are terminated on 3 min/4GB resource limit.

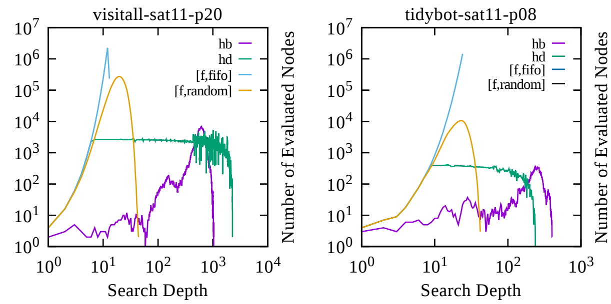

Figure 4: Distribution of the evaluated nodes per depth.

	h	hb	hd	*ro*
ipc2014 sum	14	15	**22**	15
hiking	2	2	**7**	2
tetris	0	1	**3**	1
ipc2011 sum	30	48	**50.8**	35
pegsol	17	18.5	**19**	17
scanalyzer	4	4	**6**	4
sokoban	3	3	**3.8**	3
tidybot	2	**17.5**	14	6
visitall	0	0	**3**	0

Table 2: Problems solved under 3 minutes/4GB RAM (average of 10 runs). Best results are in **bold**. We do not show the domains with no differences between configurations.

We plotted the depth of the nodes expanded by these algorithms on two representative runs (visitall-sat11-p20, tidybot-sat11-p08) in Figure 4. As expected, *ro* behaves similarly to BreadhFS/*fifo* (search is biased to the shallow depths) and Depth-diversification shows a flat distribution because it is specifically designed to achieve the fair allocation among depths. Compared to BreadthFS/*fifo* and *ro*, the increase of nodes-per-depth by IP-diversification is much slower, supporting our observation that IP is controlled by the least width in the search graph. Compared to Type-based diversification which shows linear nodes-per-depth, IP still exhibits exponential behavior because IP has no explicit mechanism for balancing the search efforts wrto depths. However, IP expands smaller number of nodes in the shallower region. Similar figures were obtained for other domains.

We also compared their performance on IPC instances. The results show that both (hd) and (hb) improves upon blind BreadthFS while not strictly dominating each other: (hb) shows better performance than (hd) on Tidybot domain. Comparison between *ro* and hb indicate that the blind performance of IP is better than that of *ro* in tidybot and pegsol.

4.1 Evaluation of IP-Diversification

Given the performance of blind search, IP-diversification is a good candidate for improving the performance of diversified heuristic search. We compared the performance of (h), the standard GBFS, with the combined Type-based diversification (hdD) from Sec. 3.1 as well as intra-plateau IP-diversification (hb:$[h, r_{\text{BIP}}]$), inter-plateau IP-diversification

		h^{CG}					h^{FF}				
		h	hb intra	hB inter	hbB both	hdD both	h	hb intra	hB inter	hbB both	hdD both
	total	187	187.2	206.8	*208.7*	**215.8**	192	207.8	232.9	**237.7**	223.9
IPC11 w/o duplicates	elev..	9	9.2	**12.6**	**13.3**	**9.7**	19	18.2	18.5	19.4	13.7
	nomy..	7	6.4	5.5	5.6	**15.1**	9	6.6	7.6	6.6	**17**
	parc..	20	19.6	13.7	12.4	18.7	20	20	19.9	18.9	20
	pegs..	20	20	19.7	19.8	20	20	20	20	20	20
	scan..	20	20	20	20	20	15	16.6	**19.1**	**19.1**	**18.6**
	soko..	16	15.9	15.8	15.2	**17**	19	18.6	18.5	18.4	17.4
	tidy..	16	17.3	17.5	17.5	**18.6**	16	15	16.4	16.3	**16.7**
	wood..	2	1.8	**14**	**12.8**	**7.7**	2	1.5	**14.8**	**15.7**	**7.2**
IPC14	barm..	0	0	0	0	0	0	0	**7.6**	**6.5**	**1**
	cave..	7	7.1	7	6.9	7	7	7	7	7	7.2
	chil..	1	0	0.1	0	**1.5**	0	0	0.1	0	0.3
	city..	0	0.2	**1.1**	0.4	**4.7**	0	0	**3**	**3.8**	**7.1**
	floo..	0	0	**0.5**	0.2	**2**	2	2	2.1	2	2.1
	ged	0	0	**4.8**	**4.6**	**9.7**	19	19.2	12.8	13	13.8
	hiki..	18	15.9	**18.7**	**18.8**	**19.7**	20	17.6	19.9	20	20
	main..	16	14.6	14.9	14.1	15.8	11	6.7	10	5.8	11.1
	open..	0	0.1	**2.5**	**2.4**	**0.5**	0	**15.7**	**11.7**	**14.5**	**7**
	park..	7	**10.4**	7.6	**10.9**	4.1	4	**5.4**	2.3	**4.8**	**5.7**
	tetr..	18	**19.7**	17.6	**19.4**	14.3	1	**8.6**	**7**	**11.1**	**4.9**
	thou..	5	4.9	5.2	5.2	5	8	**9.1**	**11.2**	**11**	**13.1**
	tran..	5	4.1	**6**	**7.1**	4.7	0	0	0	0	0
	visi..	0	0	**2**	**2.1**	0	0	0	**3.4**	**3.8**	0

Table 3: Number of solved instances (5 min, 4Gb RAM), mean of 10 runs. **h**: baseline GBFS. **hb/hB**: intra / inter-plateau IP diversification $[h, r_{\mathrm{BIP}}]$ and $alt([h], [r_{\mathrm{BIP}}])$, **hbB**: A combined IP configuration $alt([h, r_{\mathrm{BIP}}], [r_{\mathrm{BIP}}])$, **hdD**: $alt([h, \langle d \rangle], [\langle g, h \rangle, ro])$ (same as hdD from Table 1) . The same highlighting/coloring rules as Table 1 are applied, showing that intra/inter-plateau schemes based on IP are complementary. **bold** shows the improvements by **hdD**. Although **hbB** and **hdD** are comparable overall, per-domain comparison shows **hbB** and **hdD** are complementary.

(hB:$alt([h], [r_{\mathrm{BIP}}])$), and combined intra/inter-plateau IP diversification (hbB:$alt([h, r_{\mathrm{BIP}}], [r_{\mathrm{BIP}}])$).

Results are shown in Table 3. IP-diversification, applied to both intra- and inter-plateau exploration, resulted in improvements on both the h^{FF} and h^{CG} heuristics. Complementary effects similar to Table 1 are observed between hb and hB, and hbB outperforms both hb and hB. This provides additional empirical evidence for the hypothesis that intra/inter-plateau exploration are complementary, and that they can be combined to yield superior performance.

Overall, hbB performs comparably to hdD. However, note that some domains were improved by Type-based but not by IP (e.g. nomystery, sokoban, childsnack) or vise versa (transport, visitall). These results indicate that Type-based and IP diversification are orthogonal, addressing different diversity criteria (depth vs breadth).

5 Intra- and Inter-Plateau Diversification on a State-of-the-Art Planner

Up to this point, we have evaluated intra/inter-plateau exploration on greedy best-first search in order to cleanly isolate their effect. Next, we evaluate the combined effect of intra/inter-plateau exploration when applied to a state-of-the-art planner, the LAMA2011 configuration in the current version of FastDownward, which incorporates a number of search enhancement techniques such as lazy evaluation, multi-heuristic search and preferred operators. In order to focus on coverage, we only run the first iteration (unit-cost GBFS) of LAMA, denoted as $alt([h^{\mathrm{FF}}], pref(h^{\mathrm{FF}}), [h^{\mathrm{LC}}], pref(h^{\mathrm{LC}}))$, where h^{LC} denotes the landmark-count heuristic and $pref(X)$ denotes the preferred operator queue with sorting strategy X.

We apply the methods proposed in this paper incrementally. We first add a single exploration strategy to LAMA. (d, b) augments $[h]$ with type-based and IP diversification for intra-plateau exploration ($[h, \langle d \rangle]$ and $[h, r_{\mathrm{BIP}}]$), respectively. (D, B) incorporates diversification for inter-plateau exploration by adding $\langle g, h^{\mathrm{FF}} \rangle$ and $[r_{\mathrm{BIP}}]$ to LAMA's alternation queue, respectively. LAMA+D is equivalent to Type-LAMA (Xie et al. 2014).

Next, we combine intra/inter-plateau diversification methods: (dD) applies both changes in (d) and (D), and similarly (bB) applies both changes in (b) and (B). Finally, (db^{2}DB) incorporates all 4 methods into LAMA.

Let db denote $alt(\langle d \rangle, r_{\mathrm{BIP}})$, alternation between depth and IP based diversification for intra-plateau exploration, and let DB denote $alt(\langle g, h^{\mathrm{FF}} \rangle, r_{\mathrm{BIP}})$, alternation between type-based and IP based diversification for inter-plateau exploration. The resulting configuration, LAMA-db^{2}DB, incorporates all of the ideas proposed in this paper: $alt([h^{\mathrm{FF}}, db], pref(h^{\mathrm{FF}}), [h^{\mathrm{LC}}, db], pref(h^{\mathrm{LC}}), DB)$. This configuration alternates between type-based and IP diversification in each iteration. It allocates 1/5 of the entire search time to inter-plateau exploration (same as the frequency with which Type-LAMA selects from $\langle g, h^{\mathrm{FF}} \rangle$), so it spends 1/10 of the time on $[r_{\mathrm{BIP}}]$ and 1/10 of the time on $\langle g, h^{\mathrm{FF}} \rangle$). Adopting more sophisticated approaches for determining exploration frequency (Schulte and Keller 2014; Nakhost and Müller 2009) is a direction for future work.

Table 4 shows the number of solved instances in 5 min, 4GB RAM limit. Each single diversification improved the overall performance of LAMA except LAMA+B. For combinations of two methods (dD and bB), complementary effects by intra-/inter-plateau diversification similar to Table 1 are observed. Although LAMA+B did not result in improvement, adding B to LAMA+b resulted in larger coverage in LAMA+bB. Finally, bd^{2}BD outperformed all other methods. We observed complementary effects from dD and bB, each addressing different diversity criteria.

6 Conclusions and Future Work

In this paper, we first introduced the notion of *Intra-* and *Inter*-plateau exploration in satisficing heuristic search. While previous work on exploration focused on inter-plateau exploration, we argued that intra-plateau exploration addresses orthogonal issues, and showed that the type-based diversification framework originally developed for inter-plateau diversification could be used to unify intra- and inter-plateau diversification. We then showed empirically that these two modes of diversification have orthogonal, complementary effects when implemented as diversification strategies for GBFS, and showed that it is possible to combine intra/inter-

		Planners Based on the Latest FastDownward							
		LAMA	+d	+D	+dD	+b	+B	+bB	+db^{2}DB
	total	293.2	296.5	294.3	295.4	293.3	287.6	297.6	**304.5**
IPC11 w/o duplicates	elevators	20	19.3	19	19.2	20	19.4	19.9	19.6
	nomystery	10	9.9	**17.4**	**16.4**	9.8	10.4	9.7	**16.1**
	parcprinter	20	18.4	19.9	19.7	18.2	19.5	18.3	19.3
	pegsol	20	19	20	20	19.4	20	20	20
	scanalyzer	19	19.3	19.1	19.2	**19.5**	**19.6**	**19.5**	19.2
	sokoban	17	16.9	16.9	16.6	16.4	17	16.9	16.2
	tidybot	16	**17**	15.8	15.8	14.8	15.7	**16.5**	**16.5**
	woodwork	20	20	20	20	20	20	20	20
IPC14	barman	15	13.6	9.5	10.4	12.1	**16**	14.2	14
	cavediving	7	7	7.1	7.1	6.8	6.9	6.7	7
	childsnack	0	**9.3**	0.1	0	0.2	0.3	0.1	0
	citycar	2	1	**5.5**	**4.4**	**4.5**	**4.2**	**4.1**	**4.4**
	floortile	2	2	2.1	2	2	2	2	2
	ged	20	20	20	20	20	20	20	20
	hiking	18.5	18.7	17.5	18.7	**19.1**	17.5	**19.6**	18.8
	maintenance	1	1	**5.5**	**5.6**	1	1	1	**3.6**
	openstacks	20	20	20	20	20	20	20	20
	parking	19.1	**19.8**	16.7	18.7	**19.6**	18.1	18.7	**19.6**
	tetris	9.3	7.1	7.4	7.1	**12.4**	4.7	**15.3**	**14.2**
	thoughtful	14	**14.5**	**15.1**	**15.4**	13.1	**14.5**	12.9	**14.6**
	transport	3.3	**3.8**	2.6	**3.8**	**4.4**	3.7	**3.8**	3.5
	visitall	20	18.9	17.1	15.3	20	17.1	18.4	15.9

Table 4: Number of solved instances in 5min,4GB RAM. LAMA's sorting strategy is $alt([h^{\mathrm{FF}}], \mathit{pref}\left(h^{\mathrm{FF}}\right), [h^{\mathrm{LC}}], \mathit{pref}\left(h^{\mathrm{LC}}\right))$. For each heuristic $h = h^{\mathrm{FF}}$ and $h = h^{\mathrm{LC}}$ in LAMA, (d,b) augments $[h]$ with type-based and IP diversification for intra-plateau exploration ($[h, \langle d\rangle]$ and $[h, r_{\mathrm{BIP}}]$, respectively). (D,B) applies inter-plateau exploration by adding $\langle g, h^{\mathrm{FF}}\rangle$ and $[r_{\mathrm{BIP}}]$ to LAMA's alternation queue, respectively. D corresponds to Type-LAMA (Xie et al. 2014). (dD) includes both changes in (d) and (D), and similarly (bB) includes both changes in (b) and (B). Finally, (db^{2}DB) combines all methods: $alt([h^{\mathrm{FF}}, alt(\langle d\rangle, r_{\mathrm{BIP}})], \mathit{pref}\left(h^{\mathrm{FF}}\right), [h^{\mathrm{LC}}, alt(\langle d\rangle, r_{\mathrm{BIP}})], \mathit{pref}\left(h^{\mathrm{LC}}\right), alt(\langle g, h^{\mathrm{FF}}\rangle, r_{\mathrm{BIP}}))$. The same highlighting/coloring rules as Table 1 are applied. LAMA+db^{2}DB successfully combines improvements from 4 diversification strategies and achieved the best overall coverage.

plateau diversification, resulting in better performance than either class of strategy alone.

Next, we showed that type-based diversification is not sufficient for bias avoidance in graphs where nodes have largely varying number of neighbors, and proposed IP-diversification, a new breadth-aware diversification strategy which addresses this issue. We then showed that IP-diversification can be used as either intra- or inter-plateau exploration strategy, i.e., unlike depth-diversification and $\langle g, h\rangle$ type-based diversification which are specialized for either intra- or inter-plateau exploration, IP is a dual-mode diversification strategy.

Finally, we showed that incorporating these two new ideas (performing both intra/inter-plateau exploration, and IP-diversification) into FD/LAMA yields state-of-the-art performance on IPC benchmark instances.

While we investigated Bond-IP (BIP), the variant of Invasion Percolation which fixes random values to edges, the dual variant which fixes values on nodes is called *Site IP*. Analysis of SIP is a direction for future work. Valenzano et al. (2014, Section 4.3) evaluate a baseline, knowledge-free heuristic which assigns a random h-value to a node. By itself, this would behave similarly to the ro baseline strategy, if heuristic values are reevaluated for reopened nodes. By default, FastDownward reevaluates the heuristic value for reopened nodes.[2] However, Valenzano et al. disabled node-reopening in all their experimental configurations, which, in effect, fixes the random heuristic value for each node, so this should behave similarly to SIP.

This paper has shown that exploration strategies such as pure randomization, depth-diversification, $\langle g, h\rangle$ type-bucket diversification, and IP-diversification are strategies which can be plugged in as components in a search architecture which performs exploration within and among plateaus. In future work, other exploration strategies which have been developed for blind search such as novelty-based metrics used in Probe (Lipovetzky and Geffner 2011) and Iterated Width (Lipovetzky and Geffner 2012) could be used as components in this two-tiered exploration architecture.

References

Asai, M., and Fukunaga, A. 2016. Tiebreaking Strategies for Classical Planning Using A^* Search. In *Proceedings of AAAI*

[2] `http://hg.fast-downward.org/file/df227b467100/src/search/search_engines/eager_search.cc#l202`

Conference on Artificial Intelligence.

Barabási, A.-L. 1996. Invasion percolation and global optimization. *Physical Review Letters* 76(20):3750.

Chu, Y.-J., and Liu, T.-H. 1965. On shortest arborescence of a directed graph. *Scientia Sinica* 14(10):1396.

Edmonds, J. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B* 71(4):233–240.

Felner, A.; Kraus, S.; and Korf, R. E. 2003. KBFS: K-best-first search. *Annals of Mathematics and Artificial Intelligence* 39(1-2):19–39.

Gabow, H. N.; Galil, Z.; Spencer, T.; and Tarjan, R. E. 1986. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica* 6(2):109–122.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004), June 3-7 2004, Whistler, British Columbia, Canada*, 161–170.

Helmert, M. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.(JAIR)* 26:191–246.

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

Hoffmann, J., and Nebel, B. 2001b. The FF Planning System: Fast Plan Generation through Heuristic Search. *J. Artif. Intell. Res.(JAIR)* 14:253–302.

Imai, T., and Kishimoto, A. 2011. A Novel Technique for Avoiding Plateaus of Greedy Best-First Search in Satisficing Planning. In *Proceedings of AAAI Conference on Artificial Intelligence*.

Lelis, L. H.; Zilles, S.; and Holte, R. C. 2013. Stratified Tree Search: A Novel Suboptimal Heuristic Search Algorithm. In *AAMAS*, 555–562. International Foundation for Autonomous Agents and Multiagent Systems.

Lipovetzky, N., and Geffner, H. 2011. Searching for Plans with Carefully Designed Probes. In *Proceedings of the International Conference of Automated Planning and Scheduling(ICAPS)*.

Lipovetzky, N., and Geffner, H. 2012. Width and serialization of classical planning problems. In *ECAI*, volume 2012, 20th.

Nakhost, H., and Müller, M. 2009. Monte-Carlo Exploration for Deterministic Planning. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Prim, R. C. 1957. Shortest connection networks and some generalizations. *Bell system technical journal* 36(6):1389–1401.

Richter, S., and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *J. Artif. Intell. Res.(JAIR)* 39(1):127–177.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks Revisited. In *Proceedings of AAAI Conference on Artificial Intelligence*.

Röger, G., and Helmert, M. 2010. The More, the Merrier: Combining Heuristic Estimators for Satisficing Planning. In *Proceedings of the International Conference of Automated Planning and Scheduling(ICAPS)*.

Schulte, T., and Keller, T. 2014. Balancing Exploration and Exploitation in Classical Planning. In *Proceedings of Annual Symposium on Combinatorial Search*.

Tarjan, R. E. 1977. Finding optimum branchings. *Networks* 7(1):25–35.

Valenzano, R. A., and Xie, F. 2016. On the Completeness of BestFirst Search Variants that Use Random Exploration. In *Proceedings of AAAI Conference on Artificial Intelligence*.

Valenzano, R. A.; Schaeffer, J.; Sturtevant, N.; and Xie, F. 2014. A Comparison of Knowledge-Based GBFS Enhancements and Knowledge-Free Exploration. In *Proceedings of the International Conference of Automated Planning and Scheduling(ICAPS)*.

Wilkinson, D., and Willemsen, J. F. 1983. Invasion percolation: a new form of percolation theory. *Journal of Physics A: Mathematical and General* 16(14):3365.

Xie, F.; Müller, M.; Holte, R. C.; and Imai, T. 2014. Type-Based Exploration with Multiple Search Queues for Satisficing Planning. In *Proceedings of AAAI Conference on Artificial Intelligence*.

Xie, F.; Müller, M.; and Holte, R. 2014. Adding local exploration to greedy best-first search in satisficing planning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 2388–2394.