Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)

Analyzing and Avoiding Pathological Behavior in Parallel Best-First Search

Ryo Kuroiwa, Alex Fukunaga
Graduate School of Arts and Sciences
The University of Tokyo

Abstract

Recent work has experimentally shown that parallelization of Greedy Best-First Search (GBFS), a satisficing best-first search method, can behave very differently from sequential GBFS. In this paper, we propose a theoretical framework to compare parallel best-first search with sequential best-first search, including both suboptimal (GBFS, Weighted A*) and optimal (A*) best-first search methods. We analyze the extent to which the search behavior of existing parallel best-first search methods differ from sequential best-first search, and show that existing methods are vulnerable to pathological behavior, and that they can expand nodes which would not be expanded by sequential search under any tie-breaking policy. We also propose PUHF, a parallel best-first search which is guaranteed to expand a node only if there is some tie-breaking strategy for sequential search which expands the node.

1 Introduction

Best-First Search (BFS) algorithms, including A* (Hart, Nilsson, and Raphael 1968) and Greedy Best-First Search (GBFS) (Doran and Michie 1966) are widely used to solve difficult graph search problems. Parallelization of BFS is important both in order to speed up search time, as well as to solve harder problems using the aggregate RAM across multiple machines. In the case of A*, it has been shown experimentally that efficient parallelization is consistently achievable on many domains (Burns et al. 2010; Kishimoto, Fukunaga, and Botea 2013). However, it was recently shown experimentally that achieving parallel speedups compared to sequential GBFS is not as straightforward as for A* (Kuroiwa and Fukunaga 2019) – hash-based distribution, which was highly effective, for A*, can sometimes incur massive *search overhead* (large increase in nodes expanded compared to sequential GBFS) on standard benchmark domains when applied to GBFS. Previous work focused on experimental evaluation of parallel BFS performance, but to our knowledge, there has been little work on a theoretical characterization of the search overhead of parallel BFS.

This paper seeks to further our understanding of parallel BFS by developing a theoretical framework for characterizing the behavior of parallel BFS. We first extend the recent use of KBFS (Felner, Kraus, and Korf 2003) as a model for parallel BFS and prove the bounded optimality of KWA*, the K-parallel analogue of Weighted A* (WA*). Next, we consider bounding the search overhead of parallel BFS relative to its sequential counterpart. We define the notion of t-bounded search overhead and pathological (unbounded) behavior in parallel BFS, and show that parallel A*, WA*, and GBFS are all susceptible to pathological behavior when $Open$ and/or $Closed$ are shared among processes. We also define the notion of TB-bounded behavior, which is satisfied if parallel BFS is guaranteed not to expand any node which would not be expanded under any tie-breaking strategy by their sequential counterpart, and show that previous parallel BFS algorithms are not TB-bounded. We then investigate several approaches to ensuring t-bounded and TB-bounded behaviors in satisficing parallel BFS, including simple approaches which do not share $Open$ or $Closed$, as well as a sufficient criterion for ensuring TB-bounded behavior in GBFS, applying the recent notion of *bench transition systems* by Heusner, Keller, and Helmert (2017). We evaluate the search behavior and performance of these approaches.

2 Preliminaries and Background

State Space Topology State space topologies are defined following previous work (Heusner, Keller, and Helmert 2018), extended with state transition costs.

Definition 1. A *state space* is a 5-tuple $\mathcal{S} = \langle S, succ, cost, s_{init}, S_{goal} \rangle$, where S is a finite set of states, $succ : S \to 2^S$ is the successor function, $cost : S \times S \to \mathbb{N}_0$ is the cost function, $s_{init} \in S$ is the initial state, and $S_{goal} \subseteq S$ is the set of goal states. If $s' \in succ(s)$, we say that s' is a successor of s and that $s \to s'$ is a (state) transition. We assume $\forall s \in S_{goal}, succ(s) = \emptyset$. A *heuristic* for $\mathcal{S}$ is a function $h : S \to \mathbb{N}_0$ and $\forall s \in S_{goal}, h(s) = 0$. A *state space topology* is a pair $\langle \mathcal{S}, h \rangle$, where $\mathcal{S}$ is a state space.

We call a sequence of state $\langle s_0, ..., s_n \rangle$ a *path* from s_0 to s_n, and denote the set of paths from s to s' as $P(s, s')$. We denote the i th state in a path p as p_i and the length of p as $|p|$. A state s' is reachable from another state s when

$|P(s, s')| > 0$. The cost of a path p is defined as $cost(p) = \sum_{i=0}^{|p|-2} cost(p_i, p_{i+1})$. A *solution* of a state space topology is a path p from s_{init} to a goal state. $g^*(s)$ denotes the cost of the shortest path from s_{init} to s, $\min_{p \in P(s_{init}, s)} cost(p)$. $h^*(s)$ denotes the cost of the shortest path from s to a goal, $\min_{s' \in S_{goal}, p \in P(s,s')} cost(p)$. In particular, $h^*(s_{init})$ is denoted by f^*. A solution is optimal if its cost is equivalent to f^*. In this paper, we assume that a state space topology is solvable, i.e., at least one goal state is reachable from s_{init}, and there is no self-loop, i.e., $\forall s \in S, s \notin succ(s)$.

Several properties of state space topologies are defined:

Definition 2. A state space $\langle S, succ, cost, s_{init}, S_{goal} \rangle$ is *undirected* iff $\forall s \in S, \forall s' \in succ(s), s \in succ(s') \wedge cost(s, s') = cost(s', s)$. A state space topology $\langle \mathcal{S}, h \rangle$ is undirected iff $\mathcal{S}$ is undirected.

Definition 3. A state space $\langle S, succ, cost, s_{init}, S_{goal} \rangle$ is *unit-cost* iff $\forall s \in S, \forall s' \in succ(s), cost(s, s') = 1$. A state space topology $\langle \mathcal{S}, h \rangle$ is unit-cost iff $\mathcal{S}$ is unit-cost.

Definition 4. Given a state space $\mathcal{S} = \langle S, succ, cost, s_{init}, S_{goal} \rangle$, a heuristic h is *admissible* iff $\forall s \in S, h(s) \leq h^*(s)$. A state space topology $\langle \mathcal{S}, h \rangle$ is admissible iff h is admissible.

Definition 5. Given a state space $\mathcal{S} = \langle S, succ, cost, s_{init}, S_{goal} \rangle$, a heuristic h is *consistent* if $\forall s \in S, \forall s' \in succ(s), h(s) \leq cost(s, s') + h(s')$. A state space topology $\langle \mathcal{S}, h \rangle$ is consistent iff h is consistent.

Consistency implies admissibility, but not vice versa (Hart, Nilsson, and Raphael 1968).

Best-First Search A search algorithm takes a state space topology as input and returns its solution. Best-First Search (BFS) is a class of search algorithms. BFS algorithms uses an evaluation function $f : S \to \mathcal{R}$ and a tie-breaking strategy τ. BFS searches states in the order of evaluation function values (f-values). States with the same f-value are prioritized by τ.

Alg. 1 shows K Best-First Search (KBFS) (Felner, Kraus, and Korf 2003), a generalization of BFS using a parameter $k \in \mathcal{N}_+$. BFS is KBFS with $k = 1$. In each step, KBFS removes k states from $Open$ (line 6) and generates their successors. We call successor generation of s (line 8-18) the "expansion of s". $top(Open)$ is a state $s \in Open \setminus Closed$ with $f(s) = \min_{s' \in Open \setminus Closed} f(s')$. If there are multiple states with the lowest f-value in $Open$, $top(Open)$ is determined by τ. The solution path can be extracted recursively following $parent(s)$ to s_{init} from the goal state returned by Alg. 1. Returning $NULL$ means there is no solution.

In this paper, we focus on A* (Hart, Nilsson, and Raphael 1968), Weighted A* (WA*) (Pohl 1970), and Greedy Best-First Search (GBFS) (Doran and Michie 1966), three classes of BFS algorithms that use different evaluation functions. In A*, $f(s) = g(s) + h(s)$. If h is admissible, A* returns an optimal solution. In addition, if h is consistent, A* never reopens states in $Closed$ (line 12-15) because any expanded state s satisfies $g(s) = g^*(s)$. A* with a consistent heuristic expands all states having lower f-value than f^* and never expands any state having higher f-value than f^*. In WA*, $f(s) = g(s) + wh(s)$ where $w \geq 1$. WA* is w-suboptimal, i.e., returns a solution p which satisfies $cost(p) \leq wf^*$. If h is a consistent heuristic, WA* does not need to re-open states to return a w-suboptimal solution (Likhachev, Gordon, and Thrun 2003). In GBFS, $f(s) = h(s)$. GBFS has no bound for solution costs. In this paper, we ignore lines 12-15 for GBFS because re-opening states does not guarantee any bound.

Algorithm 1 K Best-first search

```
1:  Open ← {s_init}, Closed ← ∅
2:  parent(s_init) = NULL, g(s_init) ← 0
3:  while Open \ Closed ≠ ∅ do
4:      S' = ∅
5:      while |S'| < k ∧ Open \ Closed ≠ ∅ do
6:          S' ← S' ∪ top(Open \ Closed); Open ← Open \ {s}
7:      for s ∈ S' do
8:          if s ∈ S_goal then return s
9:          Closed ← Closed ∪ {s_i}
10:         for s' ∈ succ(s_i) do
11:             if s' ∈ Closed ∪ Open then
12:                 if g(s) + cost(s, s') < g(s') then
13:                     parent(s') ← s; g(s') ← g(s) + cost(s, s')
14:                     if s ∈ Closed then
15:                         Closed ← Closed \ {s'}; Open ← Open ∪ {s'}
16:             else
17:                 parent(s') ← s; g(s') ← g(s) + cost(s, s')
18:                 Open ← Open ∪ {s'}
    return NULL
```

Note that A*, WA*, and GBFS only specify evaluation functions. To fully define a search algorithm, a tie-breaking strategy τ is necessary. Previous research revealed that the tie-breaking strategy significantly affects search performance of A* and GBFS (Asai and Fukunaga 2017b; 2017a). Following the recent theoretical analysis of GBFS tie-breaking strategy (Heusner, Keller, and Helmert 2018), "the best-case tie-breaking strategy" and "the worst-case tie-breaking strategy" refer to tie-breaking strategies with which BFS expands the smallest/largest number of states in a given state space topology, respectively.

3 Analysis of Parallel Best-First Search

3.1 Parallel BFS as K-Best-First Search

Theoretical analysis of parallel best-first search is nontrivial, because parallel threads of execution often results in nondeterministic behavior (e.g., in one execution, thread 1 finishes expanding node A before thread 2 finishes expanding node B, and vice versa in another possible execution).

Following previous work (Kuroiwa and Fukunaga 2019), we model parallel BFS algorithms as KBFS. If $n' = \min(n, |Open \setminus Closed|)$ threads/processes simultaneously expand the n' best states from $Open$ in parallel BFS, the method is equivalent to KBFS. With a few idealized assumptions, KBFS can be used as a model for parallel BFS. Note that as long as a parallel BFS implementation allows *some*

execution which satisfies these assumptions, the pathologies identified below can actually occur.

Previous work modeled Hash Distributed GBFS (HDGBFS), a distributed parallel GBFS method based on HDA* (Kishimoto, Fukunaga, and Botea 2013), as KGBFS (KBFS with $f(s) = h(s)$) (Kuroiwa and Fukunaga 2019). HDGBFS distributes states among n processes according to a hash function H. In HDGBFS with n processes, each process i has $Open_i$ and $Closed_i$, and extracts the lowest f-value state s_i from $Open_i$. $s'_i \in succ(s_i)$ is sent to process $j = H(s'_i) \bmod n$ and added to $Open_j$ if $s'_i \notin Closed_j$. At the beginning of search, $Open_i = \{s_{init}\}$ if $i = H(s_{init}) \bmod n$ and otherwise $Open_i = \emptyset$. $Closed_i = \emptyset$ in all the processes. HDGBFS terminates immediately after finding a solution. If we assume that: (1) all processes are synchronized so that they all perform each local expansion simultaneously, (2) the hash function H used by HDGBFS is ideal, such that at each expansion step, the node with the (global) i-th smallest h-value is in $Open_i$, and (3) communications are instantaneous. then HDGBFS with n processes can be modeled as KGBFS with $k = n$.

K-Parallel BFS (KPBFS) (Vidal, Bordeaux, and Hamadi 2010) is a multicore parallel BFS. As with HDGBFS, we model KPGBFS (KPBFS with $f(s) = h(s)$) as KGBFS. In KPBFS, all threads share single $Open$ and $Closed$ data structures using a mutex. Each thread locks $Open$ to remove a state s with the lowest f-value in $Open$, locks $Closed$ to check duplicates and add s to $Closed$, and locks $Open$ to add $succ(s)$ to $Open$. If we assume that (1) all expansions take the same amount of time, and (2) lock wait times are negligible, then KPGBFS with n threads can be modeled as KGBFS with $k = n$ because n' threads simultaneously expand the n' lowest f-value states at every time step.

As with HDGBFS, HDA* (Kishimoto, Fukunaga, and Botea 2013) can also be modeled as KBFS. However, to guarantee optimality, HDA* continues search after finding a solution p until there is no unexpanded state with $f(s) < cost(p)$. In KA*/KWA* (KBFS with $f(s) = g(s) + h(s)$/$f(s) = g(s) + wh(s)$), we can also guarantee optimality/bounded suboptimality with a similar method. In the following theorem, we prove w-suboptimality of KWA* because KA* is a special case of KWA* with $w = 1$. We use a lemma originally used to prove the optimality of A* (Hart, Nilsson, and Raphael 1968), which also holds for KBFS.

Lemma 1. For any states $s \notin Closed$ and for any optimal path p from s_{init} to s, there exists a state $s' \in Open \setminus Closed$ on p with $g(s') = g^*(s')$.

Theorem 1. In an admissible state space topology, KWA* with $w \geq 1$ is w-suboptimal if it continues search after finding a solution p until there is no state s with $f(s) < cost(p)$ in $Open$.

Proof. Assume that $cost(p) > wf^*$ and there is no state s with $f(s) < cost(p)$ in $Open$. At least one state on an optimal path from s_{init} to a goal is not expanded and not in $Closed$. By Lemma 1, $\exists s \in Open \setminus Closed$ with $g(s) = g^*(s)$ on the optimal path. Since $f(s) = g^*(s) + wh(s)$, $f(s) \leq w(g^*(s)+h(s))$. By admissiblity, $f(s) \leq w(g^*(s)+h(s)) \leq w(g^*(s)+h^*(s)) \leq wf^* < cost(p)$, contradicting the assumption that $\nexists s$ with $f(s) < cost(p)$. □

With this modification, we model HDA* and KPA* (KPBFS with $f(s) = g(s) + h(s)$) as KA* and HDWA* (HDA* with $f(s) = g(s) + wh(s)$) and KPWA* (KPBFS with $f(s) = g(s) + wh(s)$) as KWA*. A similar theorem has been proven for Parallel Best-Nblock-First (PBNF), a multicore parallel BFS method (Burns et al. 2010).

3.2 Parallel vs. Sequential Best-First Search

We analyze how differently parallel and sequential BFS behave on the same state space topology. First, we define a quantitative measure to compare different search algorithms.

Definition 6. Given a state space topology $\mathcal{T}$, a search algorithm A is *t-bounded* relative to search algorithm B on $\mathcal{T}$ iff A performs no more than t-times as many expansions as B. A is *t-pathological* to search algorithm B on $\mathcal{T}$ iff A is not t-bounded relative to B on $\mathcal{T}$.

A is t-bounded relative to B iff A is t-bounded relative to B for all state space topologies. A is t-bounded relative to B on state space topologies with the property P iff A is t-bounded relative to B for all state space topologies with P.

A is *pathological* relative to B iff for all $t > 0$ there exists a state space topology $\mathcal{T}$, such that A is t-pathological relative to B on $\mathcal{T}$. A is pathological relative to B in state space topologies with property P iff for all $t > 0$ there exists a state space topology $\mathcal{T}$ with property P such that A is t-pathological relative to B on $\mathcal{T}$.

Intuitively, if A is t-bounded relative to B, there is a bound t on the dissimilarity between A and B. In contrast, if A is pathological relative to B, A sometimes behaves completely differently from B.

Note that bounded behavior relative to a sequential BFS does not necessarily imply lower search overhead (and hence better search performance) than an algorithm without bounded behavior. However, bounded behavior guarantees a worst-case bound on the "regret" for using a parallel BFS instead of the sequential BFS, whereas unbounded algorithms offer no such guarantee.

We now investigate whether KBFS algorithms are bounded or pathological relative to sequential BFS algorithms.

First, we show that KA* with any tie-breaking strategy is bounded relative to A* on consistent state space topologies.

Lemma 2. Given a consistent state space topology $\langle \mathcal{S}, h \rangle$, KA* with any tie-breaking strategy expands a state s with the lowest f-value in $Open \setminus Closed$ at each k expansions and s satisfies $g(s) = g^*(s)$.

Proof. The first expanded state s in each k expansions has the lowest f-value in $Open \setminus Closed$. If $g(s) > g^*(s)$, there is a state s' in $Open \setminus Closed$ which is on an optimal path from s_{init} to s and satisfies $g(s') = g^*(s')$ by Lemma 1. By consistency, $h(s') \leq h(s) + c$ where c is the optimal path cost from s' to s. Adding $g^*(s')$ to both sides, we get $f(s') \leq g^*(s) + h(s)$. Because $g^*(s) < g(s)$,

$f(s') \leq g^*(s) + h(s) < f(s)$ and this contradicts the fact that $f(s)$ is the lowest in $Open \setminus Closed$. □

Lemma 3. Given a consistent state space topology $\langle \mathcal{S}, h \rangle$ with a set of states S, let $S_l = \{s \in S \mid g^*(s) + h(s) < f^*\}$. $\min_{s \in Open \setminus Closed} f(s) \geq f^*$ holds after at most $k|S_l|$ expansions by KA* with any tie-breaking strategy.

Proof. By Lemma 2, KA* expands a state s with $f(s) = \min_{s' \in Open \setminus Closed} f(s')$ and $g(s) = g^*(s)$ at each k expansions. If $\min_{s' \in Open \setminus Closed} f(s') < f^*$, $s \in S_l$ since $f(s) = g^*(s) + h(s) < f^*$. Thus, KA* expands at least one state $s \in S_l$ at each k expansions as long as $\min_{s' \in Open \setminus Closed} f(s') < f^*$, and s will never be re-expanded since $g(s) = g^*(s)$. To guarantee w-suboptimality (Theorem 1) KA* does not terminate immediately after finding a goal. By consistency, once $\min_{s' \in Open \setminus Closed} f(s') \geq f^*$ holds, $\min_{s' \in Open \setminus Closed} f(s') < f^*$ will never hold. Since s will never be re-expanded, $\min_{s \in Open \setminus Closed} f(s) \geq f^*$ holds after at most $k|S_l|$ expansions. □

Lemma 4. Given a consistent state space topology $\langle \mathcal{S}, h \rangle$ with a set of states S, let $S_e = \{s \in S \setminus S_{goal} \mid g^*(s) + h(s) = f^*\}$. When $\min_{s \in Open \setminus Closed} f(s) \geq f^*$, KA* with any tie-breaking strategy finds an optimal solution after at most $k|S_e| + 1$ expansions.

Proof. If there is no state $s \in Open \setminus Closed$ with $f(s) = f^*$, by consistency KA* must have expanded a goal state s' with $f(s') = f^*$, and search terminates immediately. If KA* expands a goal s_g with $f(s_g) = f^*$, KA* returns the optimal solution because $\min_{s \in Open \setminus Closed} f(s) \geq f^*$. Otherwise, since $\min_{s' \in Open \setminus Closed} f(s') = f^*$, KA* expands at least one state $s \in S_e$ at each k expansions, and s will never be re-expanded by Lemma 2. This requires at most $k|S_e| + 1$ expansions. □

Theorem 2. KA* with any tie-breaking strategy is k-bounded relative to A* with the worst-case tie-breaking strategy on consistent state space topologies.

Proof. A* expands each state in S_l in Lemma 3 exactly once. By consistency, states in S_e are reachable from s_{init} by paths which only contain states in S_l and S_e. After expanding all states in S_l, $\min_{s \in Open \setminus Closed} f(s) \geq f^*$ holds. In the worst case, A* expands each state in S_e exactly once before expanding a goal. In total, A* with the worst-case tie-breaking strategy expands $|S_l| + |S_e| + 1$ states. Since KA* performs at most $k(|S_l| + |S_e|) + 1$ expansions, KA* with any tie-breaking strategy is k-bounded relative to A* with the worst-case tie-breaking strategy. □

On the other hand, other KBFS algorithms are not bounded relative to their sequential BFS counterparts. To prove pathology, we provide concrete state space topologies as figures, where circles represent states, rectangles represent sets of states, and edges represent state transitions. A vertical rectangle is a set of states with the same g-value. A horizontal rectangle is a set of states on a path.

Theorem 3. KA* with $k \geq 2$ and any tie-breaking strategy is pathological relative to A* with any tie-breaking strategy on unit-cost, undirected, and admissible but inconsistent state space topologies.

Proof. Fig. 1 is a state space topology with parameter $t > 0$. This state space topology is unit-cost, undirected, and admissible but inconsistent because $\forall i, h(s_2) > h(s_4^i) + cost(s_2, s_4^i)$. While A* expands 5 states s_0, s_1, s_3, s_5, s_7, KA* expands more than $5t$ states including $s_0, s_1, s_2, s_4^0, ..., s_4^{5t}$. Therefore KA* is t-pathological relative to A* in this state space topology. For any $t > 0$, we can create this kind of state space topology. □

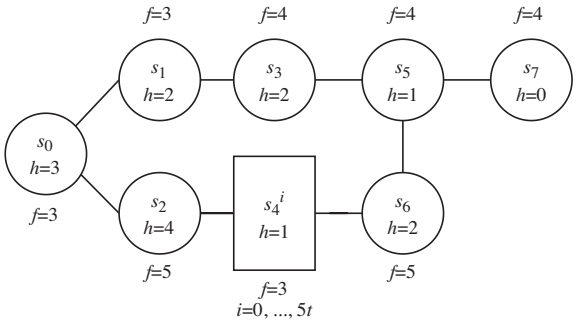

Figure 1: A state space topology with parameter $t > 0$, where $s_{init} = s_0$, $S_{goal} = \{s_7\}$, and $\forall 0 \leq i \leq 5t$, $succ(s_4^i) = \{s_2, s_6\}$. Each transition is undirected and has cost 1.

Similarly, KWA* with $w > 1$ is pathological.

Theorem 4. With weight $w > 1$, KWA* with $k \geq 2$ and any tie-breaking strategy is pathological relative to WA* with any tie-breaking strategy on unit-cost, undirected, admissible, and consistent state space topologies.

Proof. Fig. 2 is a unit-cost, undirected, admissible, and consistent state space topology with parameters $m \geq 2$ and $t > 0$. Because $f(s_3) < f(s_2^0) \wedge \forall i, f(s_1^i) < f(s_2^0)$, WA* first expands $s_0, s_1^0, ..., s_1^{m-2}, s_3$ regardless of m. Setting $m > \frac{w}{w-1}$, $f(s_9) < f(s_7) < f(s_5) < f(s_2^0)$ holds, so WA* expands s_5, s_7, s_9 and terminates. In total, WA* expands $m + 4$ states. On the other hand, KWA* simultaneously expands s_1^i and s_2^i. After expanding s_3 and s_4, KWA* must expand $s_6^0, ..., s_6^{(m+4)t}$ because $\forall j, f(s_6^j) < f(s_5) < f(s_8)$. Since KWA* expands more than $(m + 4)t$ states, KWA* is t-pathological relative to WA* in this state space topology. For any $t > 0$, we can create this kind of state space topology. □

Setting $m = 2$ in Fig. 2, we can also show the pathology of KGBFS. We omit the proof because it is almost the same as Theorem 4.

Theorem 5. KGBFS with $k \geq 2$ and any tie-breaking strategy is pathological relative to GBFS with any tie-breaking strategy on unit-cost, undirected, admissible, and consistent state space topologies.

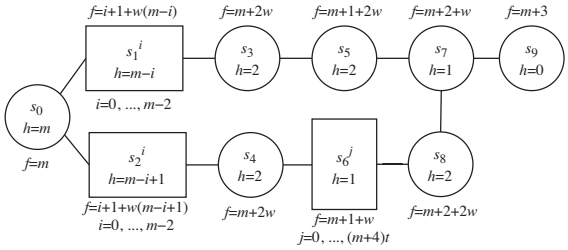

Figure 2: A state space topology with parameter $t > 0$ and $m \geq 2$, where $s_{init} = s_0$, $S_{goal} = \{s_9\}$, $succ(s_0) = \{s_1^0, s_2^0\}$, $\forall 0 < i < m-2, succ(s_1^i) = \{s_1^{i-1}, s_1^{i+1}\} \wedge succ(s_2^i) = \{s_2^{i-1}, s_2^{i+1}\}$, $succ(s_3) = \{s_1^{m-2}, s_5\}$, $succ(s_4) = \{s_2^{m-2}\} \cup \{s_6^j \mid 0 \leq j \leq (m+4)t\}$, and $\forall 0 \leq j \leq (m+4)t, succ(s_6^j) = \{s_4, s_8\}$. Each transition is undirected and has cost 1.

3.3 Pathology Due to Shared *Closed*

On the state space topologies introduced above, KBFS ignores a state expanded by sequential BFS (such as s_5 in Fig. 2) to expand states not expanded by sequential BFS (such as s_6^j in Fig. 2). Since KBFS shares a single priority queue as *Open*, once states with the lowest f-values are found, KBFS must expand all of these states. A simple modification to avoid this behavior is to use multiple *Open* structures. We call this K Independent Open BFS (KIOBFS). KIOBFS with k uses $Open_0$ to $Open_{k-1}$. At the beginning of KIOBFS, $Open_0 = \{s_{init}\}$ and $\forall i > 0$, $Open_i = \emptyset$. At each step, KIOBFS expands $\min(k, \sum_{i=0,\ldots,k-1} |Open_i|)$ states. KIOBFS removes a state s_i from each $Open_i$ if $|Open_i| > 0$. Otherwise, KIOBFS removes a state s_i with the lowest f-value among all *Open*. After removing $\min(k, \sum_{i=0,\ldots,k-1} |Open_i|)$ states, KIOBFS adds $succ(s_i)$ to $Open_i$. In Figs. 1 and 2, KIOBFS is k-bounded relative to sequential BFS because $Open_0$ behaves in the same way as *Open* in sequential BFS.

However, in general, parallel GBFS using KIOBFS (KIOGBFS) is pathological relative to sequential GBFS because of the shared *Closed*.

Theorem 6. KIOGBFS with $k \geq 2$ and any tie-breaking strategy is pathological relative to GBFS with any tie-breaking strategy.

Proof. Fig. 3 is a state space topology with a parameter $t > 0$. GBFS expands 6 states, s_0, s_1, s_3, s_5, s_7, and s_9. KIOGBFS expands s_1 and s_2 simultaneously after expanding s_0 and adds s_3 to $Open_0$ and $s_5, \forall 0 \leq i \leq k-1, s_4^i$ to $Open_1$. KIOGBFS expands s_3 for $Open_0$ and s_5 for $Open_1$. If $k > 2$, since $\forall i > 1$, $Open_i = \emptyset$, we assume that KIOGBFS expands s_4^i for $Open_i$ with $i > 1$. After these expansions, $Open_0 = \emptyset$ because s_5, the successor of s_3, is already expanded and in *Closed*. At the next step, KIOGBFS expands s_4^0 for $Open_0$, s_4^1 for $Open_1$, and $s_6^{i,j}$ for $\forall i > 1$, $Open_i$ if $k > 2$ because $h(s_4^i) < h(s_6^{i,j}) < h(s_7) < h(s_8)$. After this step, KIOGBFS must expand $\forall 0 \leq i < k, 0 \leq j \leq 6t, s_6^{i,j}$ before expanding s_7. Therefore KIOBFS is t-pathological relative to GBFS on this state space topology. For any $t > 0$, we can create this kind of state space topology. □

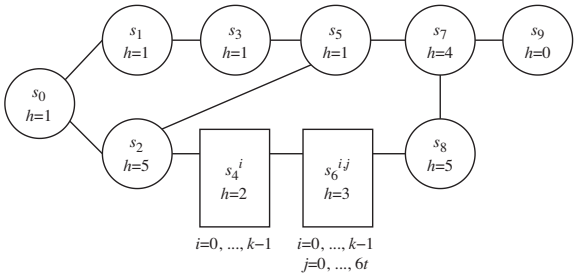

Figure 3: A state space topology with parameter $t > 0$ where $s_0 = s_{init}$, $S_{goal} = \{s_9\}$, $succ(s_2) = \{s_4^i \mid 0 \leq i < k\} \cup \{s_5\}$, $\forall 0 \leq i < k, succ(s_4^i) = \{s_2\} \cup \{s_6^{i,j} \mid 0 \leq j \leq 6t\}$, and $\forall 0 \leq i < k, \forall 0 \leq j \leq 6t, succ(s_6^{i,j}) = \{s_4^i, s_8\}$.

Thus, using multiple *Open* structures is not sufficient to avoid pathological behavior, and sharing either *Open* or *Closed* opens the possibility of pathological behavior.

3.4 TB-Bounded Behavior

Using the notion of t-boundedness, we have analyzed existing parallel BFS algorithms with respect to whether the number of expanded nodes can be bounded relative to sequential BFS. Next, we consider a different notion of bounded behavior, based on whether parallel BFS searches the same region of the search topology as its sequential counterpart.

Definition 7. Let $\mathcal{T}$ be a state space topology. A search algorithm A is *TB-bounded* relative to a search algorithm B on $\mathcal{T}$ iff A does not expand any states which are not expanded by B with any tie-breaking strategy. A is TB-bounded relative to B iff A is TB-bounded on all state space topologies.

Theorem 7. KA*, KWA*, KGBFS, and KIOGBFS with $k \geq 2$ and any tie-breaking strategy are not TB-bounded relative to their sequential counterparts on unit-cost, undirected, admissible, and consistent state space topologies.

Proof. In Fig. 4, although A*, WA*, and GBFS never expand s_2 under any tie-breaking strategy, KA*, KWA*, KGBFS, and KIOGBFS expand it. □

4 Bounded Parallel Best-First Search

As we have shown above, KWA* or KGBFS can behave arbitrarily differently from sequential WA* or GBFS. In this section, we investigate parallel GBFS methods which provide some guaranteed similarity to the search behavior of GBFS.

4.1 P_{GBFS}: Using Independent *Open* and *Closed*

P_{GBFS} is a parallel search portfolio whose components are independently executed GBFS threads with different tie-breaking strategies. All threads are completely independent and do not share *Open* and *Closed*. To our knowledge,

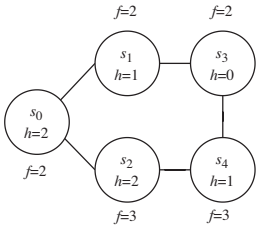

Figure 4: A state space topology where $s_0 = s_{init}$ and $S_{goal} = \{s_3\}$. Each transition is undirected and has cost 1.

the first investigation of using completely independent best-first search processes is due to Knight (1993). Recently, P_{GBFS} was evaluated for multi-core search on classical planning (Kuroiwa and Fukunaga 2019).

Using completely separate *Open* and *Closed* structures among the processes trivially eliminates pathology. P_{GBFS} with k threads is clearly both k-bounded and TB-bounded relative to GBFS if one of the threads uses the same tie-breaking strategy as sequential GBFS.

On the other hand, the threads lack the ability to explicitly exploit the work done by the other processes. Another drawback of P_{GBFS} is memory partitioning – the lack of sharing among processes also means that each process only has $1/k$ the memory available to sequential GBFS running on the same machine. This can cause P_{GBFS} to fail for memory-intensive problems which are solvable by GBFS.

4.2 Shared Evaluation Caches

While sharing *Open* or *Closed* among processes in parallel BFS can lead to pathological behavior, a safe, more limited alternative which allows some sharing of work is a shared evaluation cache. In state-of-the-art planners using BFS, a large portion of the runtime is consumed by the computation of heuristic h-values. The h-values of all states evaluated by all processes can be stored and looked up in a shared evaluation cache implemented as a lock-free hash table. If the heuristic function is path-independent, the stored, shared h-values will be valid for all processors, and the cache can potentially significantly reduce runtime if many states are evaluated by multiple processes. Furthermore, unlike sharing *Open* and *Closed*, a shared evaluation cache does not introduce the possibility of one search process influencing the node expansion order in another process, so pathology is trivially avoided. For example, we can extend P_{GBFS} to P_{GBFS}/C, which uses an evaluation cache shared among all P_{GBFS} threads, and the node expansion order in each thread will be exactly the same in P_{GBFS}/C as in P_{GBFS}.

4.3 A Sufficient Criterion for TB-Bounded Behavior in Parallel GBFS

Next, we identify a sufficient criterion for guaranteeing TB-bounded behavior in a parallel GBFS which can be modeled in our KBFS-based framework.

Applying the notion of bench transitions, recently proposed by Heusner, Keller, and Helmert in a theoretical analysis of GBFS (Heusner, Keller, and Helmert 2017; 2018), we get a necessary and sufficient criterion for TB-bounded behavior.

Definition 8. Let $\langle \mathcal{S}, h \rangle$ be a state space topology with states S and $P(s) = \{p \in P(s, s') \mid s' \in S_{goal}\}$. The high-water mark of $s \in S$ is

$$hwm(s) := \begin{cases} \min_{p \in P(s)}(\max_{s' \in p} h(s')) & \text{if } P(s) \neq \emptyset \\ \infty & \text{otherwise} \end{cases}$$

The high-water mark of a set of states $S' \subseteq S$ is defined as

$$hwm(S') := \min_{s \in S'} hwm(s)$$

Definition 9. A state s of a state space topology $\langle \mathcal{S}, h \rangle$ is a *progress state* iff $hwm(s) > hwm(succ(s))$.

Definition 10. Let $\langle \mathcal{S}, h \rangle$ be a state space topology with a set of states S. Let $s \in S$ be a progress state.

The *bench level* of s is $level(s) = hwm(succ(s))$.

The *inner bench states* $inner(s)$ for s consist of all states $s'' \neq s$ that can be reached from s on paths on which all states $s' \neq s$ (including s'' itself) are non-progress states and satisfy $h(s') \leq level(s)$.

The *bench exit states* $exit(s)$ for s consist of all progress states s' with $h(s') = level(s)$ that are successors of s or of some inner bench state of s.

The *bench states* $states(s)$ for s are $\{s\} \cup inner(s) \cup exit(s)$.

The *bench induced by* s, denoted by $\mathcal{B}(s)$, is the state space with states $states(s)$, initial state s, and goal states $exit(s)$. The successor function is the successor function of S restricted to $states(s)$ without transitions to s and from bench exit states $exit(s)$.

Definition 11. Let $\mathcal{T} = \langle \mathcal{S}, h \rangle$ be a state space topology with initial state s_{init}. The bench transition system $\mathcal{B}(\mathcal{T})$ of $\mathcal{T}$ is a directed graph $\langle V, E \rangle$ whose vertices are benches. The vertex set V and directed edges E are inductively defined as the smallest sets that satisfy the following properties:

1. $\mathcal{B}(s_{init}) \in V$
2. if $\mathcal{B}(s) \in V$, $s' \in exit(s)$, and s' is a non-goal state, then $\mathcal{B}(s') \in V$ and $\langle \mathcal{B}(s), \mathcal{B}(s') \rangle \in E$

Theorem 8. Let $\mathcal{T} = \langle \mathcal{S}, h \rangle$ be a state space topology with set of states S and bench transition system $\langle V, E \rangle$. For each state $s \in S$, it holds that $s \in \mathcal{B}(s')$ for some $\mathcal{B}(s') \in V$ iff there is a tie-breaking strategy with which GBFS expands s.

Definitions 8–11 are from Heusner, Keller, and Helmert (2018), and Theorem 8 is from Heusner, Keller, and Helmert (2017).

The bench transition system (BTS) defines the set of all nodes which are candidates for expansion by BFS with some tie-breaking strategy. If parallel BFS expands nodes which are outside of the BTS, then it is not TB-bounded by Theorem 8, and vice versa.

During search, we can identify states in *Open* which are guaranteed to be in the BTS.

Theorem 9. When s is expanded by GBFS with some tie-breaking strategy, if $h(s') \leq h(s)$ where $h(s') = \min_{s'' \in succ(s)} h(s'')$, s' is also expanded by GBFS with some tie-breaking strategy.

Proof. If s' is inserted to *Open* by GBFS, since $h(s)$ was the lowest in *Open* when s was expanded by GBFS and $h(s')$ is the lowest in $\{s\} \cup succ(s)$, $h(s')$ is the lowest in *Open* after the expansion of s. There exists a tie-breaking strategy with which GBFS expands s'. Otherwise, $s' \in Closed$ holds and s' was expanded. □

Prioritizing these states helps parallel GBFS to behave similarly to sequential GBFS. In LG, a previous distributed parallel GBFS (Kuroiwa and Fukunaga 2019), each process prioritizes a state s with the lowest h-value among states generated so far in that process. This criterion can be viewed as a stricter version of the criterion in Theorem 9. However, LG is pathological and not TB-bounded relative to GBFS on the state space in Fig. 2.

PUHF: A TB-Bounded Parallel GBFS To investigate how the criterion above can be applied in practice, we propose Parallel Under High-water mark First (PUHF), a TB-bounded parallel GBFS in Algorithm 2. Unlike P_{GBFS}, PUHF shares *Open* and *Closed* so that it can use memory space efficiently.

Algorithm 2 Parallel Under High-water mark First

```
1: parent(s_init) ← NULL; certain(s_init) ← true
2: Open ← {s_init}, Closed ← {s_init}; ∀i, s_i ← NULL
3: for i ← 0, ..., k − 1 in parallel do
4:     loop
5:         lock(Open)
6:         if ∀j, s_j = NULL then
7:             if Open = ∅ then unlock(Open); return NULL
8:             if certain(top(Open)) = false then
9:                 certain(top(Open)) ← true
10:        if certain(top(Open)) = true then
11:            s_i ← top(Open); Open ← Open \ {s_i}
12:        unlock(Open)
13:        if s_i = NULL then continue
14:        if s_i ∈ S_goal then return s_i
15:        for s'_i ∈ succ(s_i) do
16:            lock(Closed)
17:            if s'_i ∉ Closed then
18:                Closed ← Closed ∪ {s'_i}
19:                unlock(Closed)
20:                parent(s'_i) ← s_i
21:                if h(s'_i) = min_{s∈{s_i}∪succ(s_i)} h(s) then
22:                    certain(s'_i) ← true
23:                lock(Open); Open ← Open ∪ {s'_i};
   unlock(Open)
24:            else
25:                unlock(Closed)
26:        s_i ← NULL
```

PUHF is based on KGBFS, but marks states certainly included in the BTS as *certain* and only expands *certain* states. PUHF is TB-bounded relative to GBFS.

Theorem 10. PUHF is TB-bounded relative to GBFS.

Proof. Let $\mathcal{T}$ be a state space topology and its bench transition system $\mathcal{B}(\mathcal{T}) = \langle V, E \rangle$. Let $S' = \{s \in \mathcal{B}(s') \mid \mathcal{B}(s') \in V\}$. By Theorem 8, PUHF is TB-bounded iff it never expands states not in S'.

Since PUHF only expands *certain* states from *Open*, we show that all *certain* states in *Open* are in S'. The proof is by induction over states marked as *certain*. Suppose that PUHF has not expanded any state not in S'.

In line 22, since $s_i \in S'$, $h(s'_i) = \min_{s \in succ(s_i)} h(s_i)$, and $h(s'_i) \leq h(s_i)$, $s'_i \in S'$ by Theorem 9.

Let $s_{min} = top(Open)$ in line 9. s_{min} is marked as *certain* here. Since $\forall 0 \leq j \leq k-1, s_j = NULL$, no thread is expanding a state and all states that are not expanded are in *Open*. No state is removed from *Open* in lines 5–12 because *Open* is locked. If $s_{min} = s_{init}$, $s_{min} \in S'$. Assume that $s_{min} \notin S'$. p, the parent of s_{min}, is marked as *certain* since p is expanded in S' by the induction hypothesis. If p is a progress state, $h(s_{min}) > hwm(succ(p))$. Otherwise, $h(s_{min}) > hwm(succ(s'))$ where $\mathcal{B}(s')$ is the bench p belongs to. Although it is possible that p is in multiple benches, we can determine $\mathcal{B}(s')$ by the path PUHF followed from s_{init} to p. In any case, since there is a path to a goal whose high-water mark is lower than $h(s_{min})$, *Open* contains a state with lower h-value than $h(s_{min})$. By contradiction, $s_{min} \in S'$. □

SPUHF: Enhancing PUHF Using Speculation and Shared Evaluation Cache PUHF wastes CPU resources idling when no states are marked as *certain*. Instead of idling, we can use any available threads to perform *speculative* search of non-certain states. The speculative search is performed using *SOpen* and *SClosed* structures, which are distinct from the main *Open* and *Closed* structures. If the speculative search finds a solution, it is returned, but otherwise, the speculative search does not directly influence the expansion order of the non-speculative search.

Nodes are *not* shared/communicated between the speculative and non-speculative open/closed data structures, as that would require careful coordination in order to assure that the non-speculative *Open* and *Closed* structures are not corrupted by nodes that are outside the BTS. Instead, a shared evaluation cache (Sec. 4.2) is used, so that if the speculative search successfully "pre-expands" a state that would be expanded by the non-speculative search, the computation for the ensuing node evaluation computations are reused.

There are many possible policies for the speculative search. In Speculative PUHF (SPUHF), the current implementation, all states which are not *certain* are inserted into *SOpen* and *SClosed*. When a thread has to wait, the thread expands state s from *SOpen*, and its successors are inserted in *SOpen* and *SClosed*. All evaluation results are stored in a shared evaluation cache. To maximize the chances of cache hits for states which are actually in the BTS, the expansion priority policy for *SOpen*, in decreasing order of priority is: (1) successors of states expanded from *Open*, if any (according to h-value and then the tie-breaking strategy τ), (2) all other states according to h-value, then τ.

method	*Open*, *Closed*	boundedness
HDA* (Kishimoto et al., 2013)/consistent	distributed	k
HDA*/inconsistent	distributed	pathological
KWA*/$w \geq 1$ (Felner et al. (2003))	shared	pathological
KPGBFS (Vidal et al. (2010))	shared	pathological
HDGBFS (Kuroiwa and Fukunaga 2019)	distributed	pathological
LG (Kuroiwa and Fukunaga 2019)	distributed	pathological
KIOGBFS	independent, shared	pathological
P_{GBFS} P_{GBFS}/C	independent	k, TB
PUHF, SPUHF	shared	TB

Table 1: Summary of parallel BFS methods.

4.4 Summary of Parallel BFS Methods

Table 1 summarizes the theoretical contributions of this paper. We show the previous parallel BFS methods analyzed in this paper as well as the newly proposed methods.

5 Experimental Evaluation

We evaluated P_{GBFS}, P_{GBFS} with shared evaluation cache (P_{GBFS}/C), KPGBFS, PUHF, and SPUHF on 1390 instances from 55 domains from the satisficing track of IPC-98 to IPC-18. As a baseline, we used GBFS. For comparison, we also include LG. GBFS, KGBFS, PUHF, SPUHF, and LG used FIFO tie-breaking. All algorithms used the unit-cost version of the FF heuristic (Hoffmann and Nebel 2001). We implemented all algorithms in C++14 (GCC 7.4) from scratch, and used the Fast Downward parser/preprocessor (Helmert 2006). Actions are ordered according to Fast Downward translator ordering. For *Closed* and the shared evaluation cache, we used lock-free hash sets (Michael 2002). Thus, unlike pseudocode in Algorithm 2, locking *Closed* is unnecessary. All experiments were run on Amazon EC2 cloud r4.4xlarge instances (122 GiB RAM, 16 vcpus) with a memory limit of 122 GiB and a time limit of 5 minutes. Parallel algorithms used 16 threads. We run LG, KPGBFS, PUHF, and SPUHF 5 times and show the median.

Table 2 shows the number of solved instances (coverage). The shared evaluation cache significantly increases coverage in P_{GBFS} and PUHF (total coverage: P_{GBFS}/C 928 vs. P_{GBFS} 913, SPUHF 864 vs. PUHF 820).

The algorithms which do not guarantee TB-boundedness (LG, KPGBFS) performed better overall than SPUHF. However, there is no clear dominance relationship among LG, KPGBFS, and SPUHF. SPUHF had higher coverage than both LG and KPGBFS in 6 domains. Fig. 5 shows that SPUHF had better search time than KPGBFS on 194/826 solved instances, and had fewer expansions than KPGBFS on 84 instances. Also, SPUHF had better search time than LG on 559/836 instances, and had fewer expansions than LG on 314 instances (not in Fig. 5 due to space). Overall, while TB-boundedness does not result in improved performance compared to non-TB-bounded, shared *Open*/*Closed* algorithms, there does not seem to be a large penalty for ensuring TB-bounded behavior.

Comparing P_{GBFS}/C and SPUHF, which are both TB-bounded, although P_{GBFS}/C has higher overall coverage, they are somewhat complementary. SPUHF solved 28 instances from 11 domains not solved by P_{GBFS}/C, while

	GBFS	P_{GBFS}	P_{GBFS}/C	LG		KPGBFS		PUHF		SPUHF	
	cov	cov	cov	+	-	+	-	+	-	+	-
agricola	3	6	**8**	5	0	**7**	0	**7**	0	6	0
airport	31	**40**	**40**	**6**	**0**	5	**0**	3	2	5	1
caldera	8	**10**	9	**1**	**0**	0	2	**1**	**0**	**1**	**0**
caldera-split	4	4	4	**2**	0	0	0	0	0	0	0
cavediving	6	**7**	**7**	1	0	1	0	1	0	1	0
data-network	1	**7**	**7**	**2**	0	**2**	0	**2**	0	1	0
depot	13	16	**18**	2	**0**	2	1	0	**0**	**3**	**0**
driverlog	19	**20**	19	0	**0**	0	**0**	0	**0**	**1**	1
elevators	13	**17**	16	**6**	1	5	1	**6**	**0**	5	2
freecell	76	**80**	**80**	**4**	0	**4**	0	3	0	**4**	0
ged	17	**20**	**20**	**3**	0	**3**	0	0	0	**3**	0
grid	4	**5**	**5**	0	0	**1**	0	**1**	0	**1**	0
hiking	19	19	**20**	0	1	**1**	1	**1**	1	0	**0**
maintenance	9	14	**15**	**4**	2	2	3	2	**0**	3	4
mprime	29	**30**	**30**	**2**	0	1	0	1	0	1	0
mystery	17	**18**	**18**	**1**	0	0	0	0	0	0	0
nomystery	8	11	**12**	2	1	**3**	1	2	1	2	1
nurikabe	9	**14**	13	1	2	**2**	**1**	0	**1**	1	**1**
openstacks	0	**8**	6	9	0	**11**	0	0	0	0	0
organic-split	10	9	**11**	**2**	1	**2**	1	0	**0**	**2**	**0**
parcprinter	12	**17**	**17**	**8**	**0**	**8**	**0**	1	**0**	2	1
parking	0	**5**	**5**	11	0	**12**	0	5	0	6	0
pathways	10	**26**	24	2	**0**	0	2	1	**0**	**3**	**0**
pipes-notank	28	34	**35**	**5**	**0**	4	1	2	**0**	**5**	**0**
pipes-tank	22	26	**27**	2	1	**3**	1	0	2	1	**0**
rovers	21	**27**	**27**	**4**	0	2	0	**4**	0	3	0
scanalyzer	16	15	**16**	**2**	**0**	**2**	**0**	1	**0**	**2**	1
settlers	2	5	**6**	**2**	2	**2**	**1**	1	2	**2**	**1**
snake	5	5	**6**	0	0	0	0	0	0	0	0
sokoban	15	17	**19**	**4**	0	**4**	0	0	0	**4**	0
spider	7	8	**9**	4	0	5	0	1	0	**6**	0
storage	20	20	**21**	0	**0**	0	1	0	1	0	1
termes	8	11	**14**	**6**	0	**6**	0	2	0	**6**	0
tetris	2	**15**	**15**	7	0	7	0	**9**	0	8	0
thoughtful	9	15	**16**	**3**	2	2	2	0	2	0	**1**
tidybot	13	**15**	**15**	1	0	**2**	0	1	0	1	0
tpp	16	**25**	**25**	6	0	**8**	0	6	0	7	0
trucks	12	**18**	17	**5**	0	1	0	3	0	3	0
visitall	7	10	**11**	10	0	**13**	0	1	0	**13**	0
total	764	913	**928**	**137**	13	135	19	68	**12**	115	15

Table 2: 'cov' is coverage. '+'/'-' is the number of solved/unsloved instances unsolved/solved by GBFS.

P_{GBFS}/C 92 instances from 27 domains not solved by SPUHF. The large coverage gap in visitall between P_{GBFS}/C and SPUHF (11 vs. 20) shows that there is a significant advantage of using a focused, shared *Open*/*Closed* approach over a portfolio approach in some domains. Fig. 5 shows that the runtime for P_{GBFS}/C was faster than SPUHF on 449/836 instances from 48 domains, while the runtime for SPUHF was faster than P_{GBFS}/C on 387 instances from 43 domains. Also, P_{GBFS}/C expanded fewer nodes than SPUHF in 341 instances from 47 domains, while SPUHF expanded fewer nodes than P_{GBFS}/C in 492 instances from 46 domains.

6 Conclusion

This paper presented a framework for analyzing search behavior and overheads in parallel BFS relative to sequential BFS, as well as methods for achieving behavior similar to sequential BFS. Our contributions are: (1) an extended framework for analyzing the search behavior of parallel BFS based on KBFS (Felner, Kraus, and Korf 2003), (2) analysis of the search overhead of parallel A*, WA*, and GBFS based on the notion of t-bounded and pathological behavior, (3) the notion of TB-bounded parallel BFS, which only

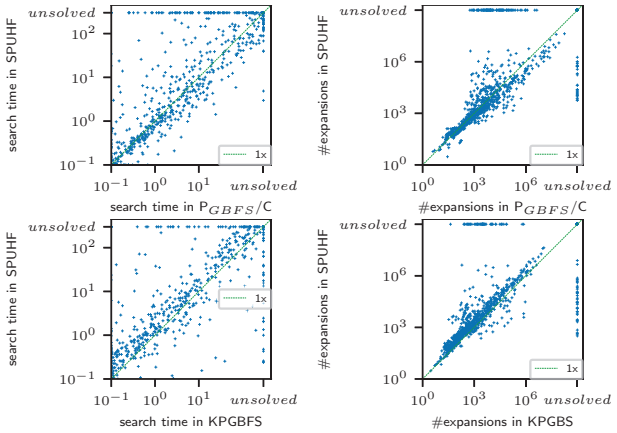

Figure 5: Comparison of different parallel GBFS methods.

searches the states expanded by sequential BFS with some tie-breaking strategy and a sufficient criterion for ensuring TB-bounded behavior of parallel GBFS, and (4) experimental evaluation of several approaches which guarantee bounded behavior, including P$_{\text{GBFS}}$/C, which achieves new state-of-the-art performance and significantly improves coverage compared to P$_{\text{GBFS}}$, LG, and KPGBFS.

Recent work on parallel BFS has focused on experimental approaches to achieving good parallel performance. Although previous work showed that the extremely large search overhead sometimes incurred by parallel GBFS can be addressed by ad hoc mechanisms such as LG (Kuroiwa and Fukunaga 2019), we developed theoretical tools which can be applied to develop parallel GBFS in a more principled manner with some theoretical guarantees. Our P$_{\text{GBFS}}$/C results show that safe methods which guarantee bounded behavior can be applied to a parallel GBFS portfolio, resulting in state-of-the-art overall coverage on IPC benchmarks.

Our results suggest that it is currently difficult to outperform portfolio-based approaches such as P$_{\text{GBFS}}$/C using methods that share *Open* and *Closed*. However, it is possible to apply our criterion for TB-boundedness to develop SPUHF, which has performance behavior somewhat complementary to P$_{\text{GBFS}}$/C. Future work will investigate integrating shared *Open*/*Closed* methods such as PUHF into portfolios to leverage this complementary behavior.

Although SPUHF currently does not achieve new state-of-the-art performance, we have shown that it is possible to guarantee TB-bounded behavior while still getting performance comparable to non-TB-bounded KPGBFS. We believe this is an important first step in a principled approach to improved shared *Open*/*Closed* algorithms. The SPUHF framework cleanly separates TB-bounded and speculative portions, so developing improved speculative policies is another direction for future work.

References

Asai, M., and Fukunaga, A. 2017a. Exploration among and within plateaus in greedy best-first search. In *Proc. ICAPS*, 11–19.

Asai, M., and Fukunaga, A. 2017b. Tie-breaking strategies for cost-optimal best first search. *J. Artif. Intell. Res.* 58:67–121.

Burns, E.; Lemons, S.; Ruml, W.; and Zhou, R. 2010. Best-first heuristic search for multicore machines. *J. Artif. Intell. Res.* 39:689–743.

Doran, J., and Michie, D. 1966. Experiments with the graph traverser program. In *Proc. Royal Society A: Mathematical, Physical and Engineering Sciences*, volume 294, 235–259.

Felner, A.; Kraus, S.; and Korf, R. E. 2003. KBFS: k-best-first search. *Ann. Math. Artif. Intell.* 39(1-2):19–39.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. on Systems Science and Cybernetics* 4(2):100–107.

Helmert, M. 2006. The fast downward planning system. *J. Artif. Intell. Res.* 26:191–246.

Heusner, M.; Keller, T.; and Helmert, M. 2017. Understanding the search behaviour of greedy best-first search. In *Proc. SOCS*, 47–55.

Heusner, M.; Keller, T.; and Helmert, M. 2018. Best-case and worst-case behavior of greedy best-first search. In *Proc. IJCAI*, 1463–1470.

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

Kishimoto, A.; Fukunaga, A.; and Botea, A. 2013. Evaluation of a simple, scalable, parallel best-first search strategy. *Artif. Intell.* 195:222–248.

Knight, K. 1993. Are many reactive agents better than a few deliberative ones? In *Proc. IJCAI*, 432–437.

Kuroiwa, R., and Fukunaga, A. 2019. On the pathological search behavior of distributed greedy best-first search. In *Proc. ICAPS*, 255–263.

Likhachev, M.; Gordon, G. J.; and Thrun, S. 2003. Ara*: Anytime a* with provable bounds on sub-optimality. In *Proc. Advances in Neural Inf. Processing Systems*, 767–774.

Michael, M. M. 2002. High performance dynamic lock-free hash tables and list-based sets. In *Proc. SPAA*, 73–82. ACM.

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artif. Intell.* 1:193–204.

Vidal, V.; Bordeaux, L.; and Hamadi, Y. 2010. Adaptive k-parallel best-first search: A simple but efficient algorithm for multi-core domain-independent planning. In *Proc. SOCS*, 100–107.