New Results in Bounded-Suboptimal Search

Maximilian Fickert,[*1] **Tianyi Gu,**[*2] **Wheeler Ruml**[2]

[1] Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[2] Department of Computer Science, University of New Hampshire, USA
fickert@cs.uni-saarland.de, {gu, ruml}@cs.unh.edu

Abstract

In bounded-suboptimal heuristic search, one attempts to find a solution that costs no more than a prespecified factor of optimal as quickly as possible. This is an important setting, as it admits faster-than-optimal solving while retaining some control over solution cost. In this paper, we investigate several new algorithms for bounded-suboptimal search, including novel variants of EES and DPS, the two most prominent previous proposals, and methods inspired by recent work in bounded-cost search that leverages uncertainty estimates of the heuristic. We perform what is, to our knowledge, the most comprehensive empirical comparison of bounded-suboptimal search algorithms to date, including both search and planning benchmarks, and we find that one of the new algorithms, a simple alternating queue scheme, significantly outperforms previous work.

Introduction

Heuristic search methods are widely used in many real-world autonomous systems. For example, search-based methods have been used in self-driving cars (Ferguson, Howard, and Likhachev 2008), Mars rovers (Mudgal et al. 2005), container terminals (Kim and Park 2004), and Amazon warehouses (Li et al. 2021), as well as for power scheduling (Thiébaux et al. 2013) and protein design (Allouche et al. 2019). However, users always want to solve search problems that are too large to be solved optimally. To find an optimal solution, one must examine every node that is not provably too expensive, that is, all reachable nodes n where $f(n) < C^*$, the optimal solution cost. Even with an almost-perfect heuristic, there can be many such nodes (Helmert and Röger 2008). This motivates suboptimal search, in which one is allowed to return a solution whose cost is greater than C^*.

One popular suboptimal setting is bounded-suboptimal search, where we require that the returned solution's cost lies within a given factor w of C^*. The best-known algorithm for this setting is weighted A* (Pohl 1970), a simple modification of A* that performs a best-first search on $f'(n) = g(n) + w \cdot h(n)$. The state-of-the-art algorithms for bounded-suboptimal search are EES (Thayer and Ruml 2011) and DPS (Gilon, Felner, and Stern 2016). Briefly, EES tries to exploit an inadmissible cost-to-go heuristic $\hat{h}(n)$ and an estimate of state space graph distance-to-go $\hat{d}(n)$ in order to quickly find solutions within the suboptimality bound. DPS tries to expand the node that is estimated to be most likely to lead to a solution within the suboptimality bound. Neither algorithm consistently outperforms the other on the classic search benchmarks on which they have been tested.

In this paper, we consider two additional ideas that have been investigated in other contexts: explicitly estimating the probability that a node will lead to a sufficiently good solution (Fickert, Gu, and Ruml 2021) and using alternating search queues ordered by different criteria (Röger and Helmert 2010). By remixing these ideas with previous work, we develop several new bounded-suboptimal algorithms. We perform the most comprehensive empirical comparison of bounded-suboptimal search algorithms to date, considering not only classic search benchmarks but also domain-independent classical planning domains, and we find that one of the new algorithms, a simple alternating queue scheme, significantly outperforms both EES and DPS across most benchmarks. In addition to establishing a new state-of-the-art in bounded-suboptimal search, this work shows that there is still plenty of room for incorporating new ideas into classic single-agent unidirectional search.

Background

A bounded-suboptimal search problem can be defined as a six-tuple $\langle S, s_{init}, succ(s), c(s, s'), G, w \rangle$, where S is a set of states, $s_{init} \in S$ is the initial state, $succ(s)$ is the transition function yielding the set of successor states of s, $c(s, s')$ is the cost of the transition from state s to its successor $s' \in succ(s)$, $G \subseteq S$ is the set of goal states, and w is a given suboptimality bound. The task is to find a path from s_{init} to a goal state, known variously as a *solution* or a *plan*, whose sum of transition costs is within the factor w of optimal. We say that a solution is w-admissible if it satisfies this requirement and an algorithm is called a bounded-suboptimal search algorithm if it is guaranteed to return a w-admissible solution.

A node n represents a state in the search space with a corresponding path to it from the initial state; the cost of the path is denoted by $g(n)$ and $h(n)$ is a heuristic estimate of

[*] These authors contributed equally.

the path cost from the state represented by n to a goal (i.e, cost-to-go). Let $h^*(n)$ be the cost of an optimal plan from n. If $h(n) \leq h^*(n)$ holds on all states then h is called admissible. In that case, $f(n) = g(n) + h(n)$ is a lower bound on the cost of any plan to a goal with prefix n. Let f_{min} be the lowest f value among all open nodes and let $best_f$ be a node with such an f value. Note that f_{min} represents a lower bound on C^*. Let $d(n)$ denote a *distance* estimate, i.e., number of state transitions to reach a goal state from n. Such an estimate can typically be derived similarly to h by considering all actions as having a cost of one. Some algorithms also use inadmissible (but potentially more accurate) estimates $\hat{h}$ and $\hat{d}$. In our experiments below, we follow prior work and derive $\hat{h}$ and $\hat{d}$ by debiasing h and d using online observations of the one-step heuristic error (Thayer, Dionne, and Ruml 2011).

Bounded-Suboptimal Search

The bounded-suboptimal setting has received attention since the earliest days of heuristic search; we review here only the work most relevant to the current state-of-the-art.

Focal Search is a family of algorithms that was first introduced with A^*_ϵ (Pearl and Kim 1982). Similar to any best-first search, focal search maintains a queue of open nodes: those that have been generated but not yet expanded. In addition, focal search also maintains a queue called focal that contains the subset of all open nodes that satisfy $f(n) \leq w \cdot f_{min}$. Because f_{min} is a lower bound on C^*, when a goal node n from focal is expanded, we have $f(n) = g(n) \leq w \cdot f_{min} \leq w \cdot C^*$, and thus n is guaranteed to satisfy the suboptimality bound. This guarantee grants the algorithm designer significant freedom for the node ordering in focal. In A^*_ϵ, focal is sorted by $d(n)$, with the intuition that we aim to find a goal in as few expansions as possible. As Thayer, Ruml, and Kreis (2009) discuss, this can actually lead to poor performance in practice because nodes with low d often have high f, causing children of expanded nodes to fail to qualify for focal.

Explicit Estimation Search (EES) (Thayer and Ruml 2011) is a more elaborate focal-style algorithm that uses three estimates to guide its search: an admissible cost heuristic h, an inadmissible cost heuristic $\hat{h}$, and an inadmissible distance-to-go heuristic $\hat{d}$. Let $\hat{f} = g + \hat{h}$ and let $best_{\hat{f}}$ be a node with lowest $\hat{f}$, which we call $\hat{f}_{min}$. In addition to open, EES maintains a queue sorted by $\hat{f}$, giving access to $best_{\hat{f}}$, and a focal-style queue, containing only those nodes n with $\hat{f}(n) \leq w \cdot \hat{f}_{min}$, that is sorted by $\hat{d}$, giving access to $best_{\hat{d}}$. If $\hat{f}(best_{\hat{d}}) \leq w \cdot f_{min}$, EES expands $best_{\hat{d}}$, as it is expected to be closest to a goal among nodes that appear sufficiently promising. Otherwise, if $\hat{f}(best_{\hat{f}}) \leq w \cdot f_{min}$, EES expands $best_{\hat{f}}$, as it is predicted to lead to the optimal solution. This also can allow EES to raise the current $best_{\hat{f}}$ value and thus enlarge the set of candidates for $best_{\hat{d}}$. If neither $best_{\hat{d}}$ nor $best_{\hat{f}}$ meet the suboptimality guarantee, EES will expand $best_f$ to raise f_{min}, hopefully allowing it to consider $best_{\hat{d}}$ or $best_{\hat{f}}$ in a future iteration.

Dynamic Potential Search (DPS) (Gilon, Felner, and Stern 2016) aims to expand the node with the highest probability of leading to a goal within the suboptimality bound. This is estimated using 'potential,' defined as $ud(n) = (w \cdot f_{min} - g(n))/h(n)$. While potential is not explicitly a probability, Stern, Puzis, and Felner (2011) show that it yields the same ordering as the probability of leading to a feasible solution, assuming the heuristic satisfies a linear error model. DPS uses a bucket-based open list with a bucket for each (g, h) pair (which have the same potential value), and reorders the buckets whenever f_{min} changes to keep the open list sorted correctly.

DPSU (Gilon, Felner, and Stern 2017) is a variant of DPS that calculates the potential using unit-cost versions of g and h, counting each edge in the search space as one regardless of its cost. It can perform better than DPS in some domains, but in general it is surpassed by DPS and EES. We also note that, while DPS and DPSU attempt to maximize the likelihood of finding a qualified solution, they do not explicitly attempt to optimize search time.

Bounded-Cost Search

Bounded-cost search is an alternative to bounded-suboptimal search in which the search is given an absolute cost bound C, rather than the relative factor w. The search then attempts to find a plan with cost at most C as quickly as possible. From an algorithm design perspective, having an absolute cost bound can simplify things versus relying on a dynamically changing f_{min}. Bounded-Cost EES (BEES) (Thayer et al. 2012) is a variant of EES that uses only two queues: open sorted by f and focal, containing only nodes with $\hat{f}(n) \leq C$, sorted by $\hat{d}$. Ordering nodes by $\hat{f}$ is not required as the focal criterion no longer depends on $\hat{f}_{min}$ and is thus static. BEES expands $best_{\hat{d}}$ unless focal is empty, in which case it expands $best_f$.

Similarly, Potential Search (PS) (Stern, Puzis, and Felner 2011) is a variant of DPS that orders nodes by $(C - g(n))/h(n)$. Bucketing is not necessary as C is static.

Expected Effort Search (XES) (Fickert, Gu, and Ruml 2021) is a recent algorithm that orders nodes on an estimate of expected search effort $xe(n) = T(n)/p(n)$, where $T(n)$ is an estimate of the time to find a solution under n and $p(n)$ is the probability that that solution will be within the cost bound. $T(n)$ is instantiated by $\hat{d}(n)$ and $p(n)$ is derived from a probability distribution centered on $\hat{f}$ as the probability mass between a lower bound (g, or f if the heuristic is admissible) and the cost bound C.

New Algorithms

We first study several variations of EES. We then study how XES can be adapted to the bounded-suboptimal setting. The major challenge that arises is how to balance between pursuing promising nodes and raising the current bound, which depends on f_{min}. We consider a simple way of resolving this issue: merely alternating between multiple open lists. We also consider incorporating potential.

All algorithms introduced in this section are variants of focal search in the sense that they maintain an open list ordered by f, and have one or more additional queues containing only nodes n with $f(n) \leq w \cdot f_{min}$ sorted by other ordering functions. Hence, any node n that they select for expansion is guaranteed to have $f(n) \leq w \cdot f_{min} \leq w \cdot C^*$, making them sound bounded-suboptimal search algorithms.

Variations on EES

Although the design choices of EES appear reasonable, it is natural to ask if they can be improved. For example, is there a more appropriate condition for selecting nodes for the focal list? Can ideas from the bounded-cost algorithms BEES and XES be useful? In this section, we present new variants of EES that probe these questions.

EES++ We first investigate the criterion for qualifying for the focal list. In the original EES design, this is $\hat{f}(n) \leq w \cdot \hat{f}_{min}$, where $\hat{f}(n)$ is our best guess of the total cost, and $w \cdot \hat{f}_{min}$ is our best guess of the suboptimality bound. However, when a node in focal is selected for expansion, EES must first check whether it satisfies the lower bound ($\hat{f}(best_{\hat{d}}) < w \cdot f_{min}$) to maintain the suboptimality guarantee. Why put a node into focal that will not be expanded? We propose to fix this issue by introducing a variant named EES++ that uses the focal condition $\hat{f}(n) \leq w \cdot f_{min}$ instead, ensuring that every node in focal is prequalified for expansion. We can then change the first step of the search strategy to simply expand $best_{\hat{d}}$ whenever focal is not empty.

Rushed EES We next consider whether the strategy of BEES can be backported to bounded-suboptimal search. This would mean a regular open list, sorted by f, and a focal list, containing nodes with $\hat{f}(n) \leq w \cdot f_{min}$, sorted by d. Like with EES++, we also simplify the expansion strategy to always select from focal if it is not empty. Otherwise, we expand $best_f$ to raise the bound, hopefully allowing more nodes to qualify. This design emphasizes the ordering on d and hence we call it Rushed EES.

Although Rushed EES appears simpler than EES, it still requires three queues because when f_{min} changes, we need to identify nodes whose $\hat{f}$ requires that they be added or removed from focal. To avoid sorting on $\hat{f}$, we would need to change the focal condition to $f(n) \leq w \cdot f_{min}$, which would leave us with A^*_ϵ, which is known to perform poorly.

EES95 EES, and all the variants introduced above, can be seen as brittle in the sense that none of them considers the uncertainty of its estimates. Unlike DPS or XES, a node believed to be barely below the bound is treated as equally important as a node that is believed to be far below the bound. Explicitly accounting for uncertainty using belief distributions has yielded promising results recently in real-time search (Mitchell et al. 2019; Fickert et al. 2020) and bounded-cost search (Fickert, Gu, and Ruml 2021). These belief distributions can be used to explicitly estimate the probability that a node leads to a solution within the suboptimality bound (we give more details in the next subsection). We investigate whether probability estimates can be useful in EES by incorporating them into the focal condition. The new variant, called EES95, only inserts nodes into focal if it additionally believes the node to have higher than 95% probability to lead to a solution within the bound. By considering the uncertainty, the algorithm makes a more robust meta-level decision in the focal condition. As this idea is orthogonal to the one considered in EES++, we also consider a combination of the two, which we call EES95++.

Exploiting Expected Effort

We also investigate whether XES can be adapted directly for bounded-cost search. Moving from a static cost bound C to the dynamic suboptimality bound $w \cdot f_{min}$ necessitates a focal search approach, where focal contains only search nodes with $f(n) \leq w \cdot f_{min}$, ordered by the expected effort, and open is ordered by f to track f_{min}. Following XES's paradigm of explicitly taking uncertainty into account, we also model the uncertainty of the suboptimality bound. We introduce two different approaches: a straightforward adaptation called Dynamic Expected Effort Search (DXES), and a more complex variant called Considerate DXES (CDXES).

DXES XES uses a single normal distribution B_{bound} centered on $\hat{f}$ to estimate the solution cost under a search node n, and thus the probability that it is within the cost bound:

$$B_{cost}(n) \sim \mathcal{N}(\hat{f}(n), ((\hat{f}(n) - f(n))/2)^2) .$$

In DXES, we also use a second distribution B_{bound} to describe our current belief about the cost bound given by the suboptimality factor w, of which our best guess is $w \cdot \hat{f}_{min}$. The value $\hat{f}_{min}$ may change after each expansion—either due to changes in the average heuristic error (when deriving $\hat{h}$ from h with online error correction) or due to nodes being added to or removed from the open list. We record the $\hat{f}_{min}$ value after each expansion in a collection δ, and use its variance to obtain a belief distribution on the bound:

$$B_{bound} \sim \mathcal{N}(w \cdot \hat{f}_{min}, var(\delta)) .$$

The probability that a node n leads to a solution within the suboptimality bound can now be expressed as the probability that a sample from $B_{cost}(n)$ is not greater than a sample from B_{bound}. This probability $P(B_{cost}(n) \leq B_{bound})$ is equivalent to $P(B_{bound} - B_{cost}(n) \geq 0)$, and we can construct the distribution $B(n) = B_{bound} - B_{cost}(n)$ by subtracting the means and adding the variances:

$$B(n) \sim \mathcal{N}(\mu = w \cdot \hat{f}_{min} - \hat{f}(n),$$
$$\sigma^2 = (\frac{\hat{f}(n) - f(n)}{2})^2 + var(\delta)).$$

From this distribution, we can compute the probability mass that is greater or equal to zero to obtain the probability that n leads to a solution within the bound (similar to XES).

Note that this approach requires fast access to three search nodes at each expansion, namely the ones with minimal expected effort, f value, and $\hat{f}$ value respectively. Accordingly, our implementation uses three queues even though DXES always expands the node $best_{xe}$ with minimal expected effort xe_{min}, which will always exist as the queue must at least contain $best_f$.

CDXES DXES focuses on finding a solution within the current known lower bound, however, it may be useful to raise the bound as well; expanding $best_f$ until f_{min} raises sufficiently such that more promising nodes (that are currently outside $w \cdot f_{min}$) become available in the focal list. We now introduce Considerate DXES (CDXES), which carefully considers the expected effort required to raise f_{min} through successive expansions of $best_f$ in order to make a node more promising than $best_{xe}$ available to focal. For example, consider a node n that is not in focal with $xe(n) = 10$ expansions when $xe(best_{xe}) = 20$. If raising f_{min} sufficiently to include n in focal takes fewer than 10 expansions, then it would be worth doing that so we can expand n afterwards and still expect less total search effort.

We estimate the effort to raise f_{min} to a desired f value $f_\circ$ (i.e., the number of required $best_f$ expansions) by

$$T_{f_\circ} = \sum_{f=f_{min}}^{f_\circ - 1} \#open_f \cdot \frac{f_\circ - f}{\epsilon_h},$$

where $\#open_f$ is the number of open nodes with the given f value, and ϵ_h is the mean one-step error in h (Thayer, Dionne, and Ruml 2011). The one-step heuristic error estimates how the f value increases on average for each expansion. For example, if $\epsilon_h = 0.2$, we can expect f to increase by one after five expansions. $T_{f_\circ}$ takes into account both the amount by which f_{min} must increase to $f_\circ$ as well as the number of open nodes with each intermediate f value. In order to express the expected effort in expansions as well, we change the remaining time estimator in $xe(n)$ from $T(n) = \hat{d}(n)$ to $T(n) = \hat{d}(n) \cdot delay$, where $delay$ is the average expansion delay (Dionne, Thayer, and Ruml 2011). Now we can consider whether it would be beneficial to expand $best_f$ in order to raise f_{min}: If there is a node n that is currently not in the focal list that has $xe(n) + T_{f(n)} < xe_{min}$, then CDXES expands $best_f$ instead of the usual $best_{xe}$.

A Simple Round-Robin Scheme

EES and CDXES attempt to carefully estimate when raising the bound is useful. However, even if the inference rules are well-founded, this metareasoning is based on heuristic online information that might be quite unreliable. Hence we also investigate a simpler alternative. We study a simple round-robin scheme using three queues: a focal list that can be instantiated with any evaluation function, an open list sorted by $\hat{f}$, and a cleanup list sorted by f. In order to guarantee that solutions are within the given suboptimality bound, the first two queues only contain nodes with $f(n) \leq w \cdot f_{min}$, while the cleanup list contains all open nodes. The search then simply alternates between these queues, expanding the node at the front of the current queue at each expansion.

We explore three instantiations for the focal list ordering: d, ud, and xe, resembling the main search strategies of EES, DPS, and DXES respectively. In particular, note that EES also considers expanding the node with either minimal distance, f, or $\hat{f}$, but follows a more sophisticated selection strategy between those queues.

One drawback of the round-robin scheme is that it does not converge to speedy search (best-first search on d) with increasing suboptimality bounds. This can be a useful property since it should lead to a solution the fastest when the solution quality does not matter (Wilt and Ruml 2014). Both EES and DXES satisfy that property: in both search algorithms, eventually all nodes are in the focal list, and in DXES all probability estimates become one with sufficiently large weights. DPS on the other hand does not converge to speedy search (and does not use a distance estimate at all).

Experimental Evaluation

We evaluate the introduced algorithms on both domain-independent planning[1] and classic search benchmarks[2]. On the planning side, we implemented the algorithms in Fast Downward (Helmert 2006) and use landmark cut (Helmert and Domshlak 2009) as both heuristic and distance estimator (using unit action costs). We compare the search algorithms using all unique STRIPS instances from the optimal tracks of the International Planning Competitions, for a total of 1652 instances from 48 domains. The experiments were run on a cluster with Intel Xeon E5-2660 CPUs using the Lab framework (Seipp et al. 2017), with time/memory limits of 30 minutes/4 GB. Figure 1 shows an overview of the results on the planning domains for various bounds ranging from $w = 1$ to $w = 2$. Normalized overall coverage corrects for the different numbers of instances per domain. Table 1 gives details on coverage in individual domains for $w = 1.5$; the last two rows show geometric means of search time and number of expansions on commonly solved instances.

Regarding classic search benchmarks, we evaluate the algorithms on C++ implementations of the sliding-tile puzzle, vacuum world (Thayer and Ruml 2011), pancake (Kleitman et al. 1975; Gates and Papadimitriou 1979; Heydari and Sudborough 1997), and racetrack (Barto, Bradtke, and Singh 1995) domains. We consider three variants of the 15-puzzle with different action costs: uniform, heavy (costs equal to the face of the tile), and inverse (one divided by the face of the tile), using the Manhattan distance as the heuristic; we use Korf's (1985) classic 100 instances.

In vacuum world, we again consider two different cost functions (uniform and heavy), using the minimum spanning tree heuristic (Thayer and Ruml 2011). We evaluate the algorithms on 60 random solvable instances of size 200×200 where cells have a 35% probability of being blocked, and the robot and dirt piles are placed at random locations (10 piles in the unit-cost version, 6 piles for heavy costs).

We also consider unit- and heavy-cost versions of the pancake problem. In the latter, the action cost of the operator that flips a prefix is the maximum among the two elements on the extreme sides of the prefix. We use the GAP heuristic (Helmert 2010) and the weaker GAP–2 heuristic that ignores the first 2 pancakes of the target state. We use 100 randomly generated instances.

Finally, we consider three different maps of the racetrack domain: Barto (Barto, Bradtke, and Singh 1995) and two

[1]https://github.com/fickert/fast-downward-xes

[2]https://github.com/gtianyi/boundedSuboptimalSearch

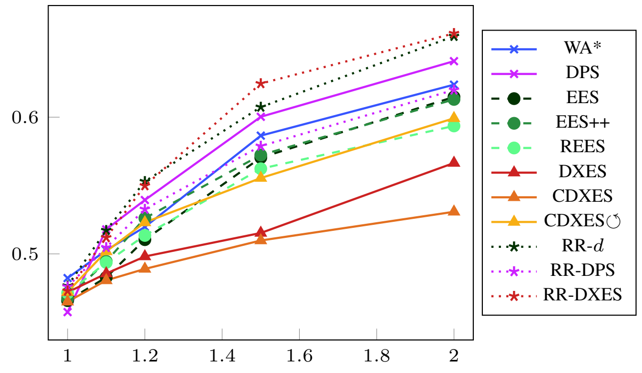

Figure 1: Normalized coverage over increasing suboptimality bounds on the planning domains.

representative maps (ost003d and den520d) from a pathfinding repository (Sturtevant 2012). The (admissible) heuristic computes the maximum distance in one dimension divided by the maximum achievable velocity, and we use a precomputed Dijkstra distance estimate. For the Barto map, we use 25 random start locations (with at least 90% of the maximum goal distance) for our evaluation. For ost003d and den520d, we use a subset of the scenarios from the repository: we choose 100 instances that have an optimal 2D path length between 50 and 100.

The results on the search domains are shown in Figures 2 and 3. They show CPU time in seconds as a function of the suboptimality bound. The error bars show 95% confidence intervals on the mean across the commonly solved instances (number shown in legend). The legends are sorted by the geometric mean across all suboptimality bounds. The right bottom panel shows aggregated results, where we compute the geometric mean of the geometric means for each domain at every suboptimality bound. We do not include inverse tiles in the aggregation as it requires different suboptimality bounds. We treat different cost functions as different domains for tiles, vacuum world, and pancake. We treat the three racetrack maps as one domain using the geometric mean.

Variations on EES

The results in Figure 2 suggest that all the variants that consider expanding nodes from three queues—EES, EES++, EES95 and EES95++—are better than REES, which only expands nodes from two queues and considers $best_f$ only when focal is empty. The weakness of REES is also observed in the planning domains (cf. Figure 1). This implies that, in bounded-suboptimal search, it is not a good idea to concentrate too much on finding a solution quickly; one must also allocate sufficient effort to bound estimation.

Looking at the different cost functions of the sliding tile domain, EES++ outperforms the original EES as problems get harder, with the most obvious data points in inverse tiles. Keeping unqualified nodes out of focal is beneficial for harder problems. For other relatively easier domains, EES++ does not make a big difference. EES95 and EES95++ perform similarly to EES, suggesting that incorporating uncertainty is not as straightforward as in the bounded-cost search setting. In fact, the aggregated plot suggests that the original EES performs the most robustly of all.

Coverage	WA*	EES	DPS	DXES	CDXES↺	RR-d	RR-DPS	RR-DXES
Airport (50)	31	30	**33**	29	32	**33**	32	**33**
Blocks (35)	33	31	34	28	31	34	32	**35**
DataNetwork (20)	14	14	14	14	14	**16**	13	**16**
Depot (22)	9	10	10	8	9	**11**	8	**11**
DriverLog (20)	**15**	**15**	**15**	14	**15**	**15**	**15**	**15**
Elevators (30)	23	21	23	21	22	**28**	23	**28**
Floortile (40)	22	17	**23**	12	16	21	21	22
Freecell (80)	15	16	15	15	17	17	15	**19**
GED (20)	15	15	**19**	15	15	15	**19**	15
Grid (5)	2	2	2	2	2	**3**	2	**3**
Gripper (20)	**20**	15	**20**	8	9	9	16	18
Hiking (20)	10	10	10	9	10	**12**	10	11
Logistics (63)	**48**	42	**48**	43	42	41	45	44
Mprime (35)	22	22	**23**	22	21	22	**23**	**23**
Mystery (19)	**17**	**17**	**17**	15	15	**17**	**17**	**17**
Nomystery (20)	**20**	**20**	**20**	14	17	**20**	**20**	**20**
Openstacks (80)	36	34	36	33	33	37	31	**38**
OrgSynth-split (20)	**16**	**16**	**16**	15	**16**	**16**	**16**	**16**
Parcprinter (30)	**30**	**30**	**30**	**30**	**30**	**30**	28	**30**
Parking (40)	18	18	20	7	13	22	17	**23**
Pathways (30)	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**
Pegsol (36)	35	33	35	33	34	**36**	35	**36**
Pipes-notank (50)	23	24	23	17	24	26	24	**27**
Pipes-tank (50)	13	13	13	14	14	17	13	**18**
PNetAlignment (20)	11	11	11	8	9	**13**	10	**13**
PSR (50)	49	49	**50**	49	**50**	**50**	**50**	**50**
Rovers (40)	14	12	13	11	15	14	14	**18**
Satellite (36)	**14**	13	**14**	11	13	13	**14**	12
Scanalyzer (28)	17	12	**20**	12	15	16	17	18
Snake (20)	7	7	7	6	**8**	7	7	7
Sokoban (30)	**29**	28	**29**	28	28	**29**	**29**	**29**
Spider (20)	**12**	11	**12**	11	11	**12**	11	**12**
Storage (30)	16	17	16	15	17	**18**	16	17
Termes (20)	7	**11**	9	6	6	10	7	10
Tetris (17)	7	7	**8**	6	6	**8**	7	**8**
Tidybot (30)	21	21	21	19	22	23	20	**26**
TPP (30)	7	8	8	8	**9**	**9**	7	8
Transport (59)	18	19	18	18	20	22	18	**23**
Trucks (30)	**21**	18	18	14	18	20	**21**	19
VisitAll (33)	21	20	21	18	21	**23**	21	**23**
Woodworking (30)	26	27	26	26	28	29	27	**30**
Zenotravel (20)	14	14	**15**	13	13	14	14	14
Others (274)	**191**	**191**	**191**	**191**	**191**	**191**	**191**	**191**
Sum (1652)	995	967	1012	894	957	1025	982	**1052**
Normalized (%)	58.7	57.0	60.0	51.5	55.6	60.7	57.9	**62.5**
Expansions	569	558	472	734	511	383	665	**371**
Search time (s)	0.65	0.91	**0.55**	1.09	0.83	0.65	1.05	0.65

Table 1: Coverage over the IPC instances for $w = 1.5$.

Exploiting Expected Effort

Consider Figure 1. Somewhat surprisingly, the success of XES in bounded-cost search does not directly transfer to

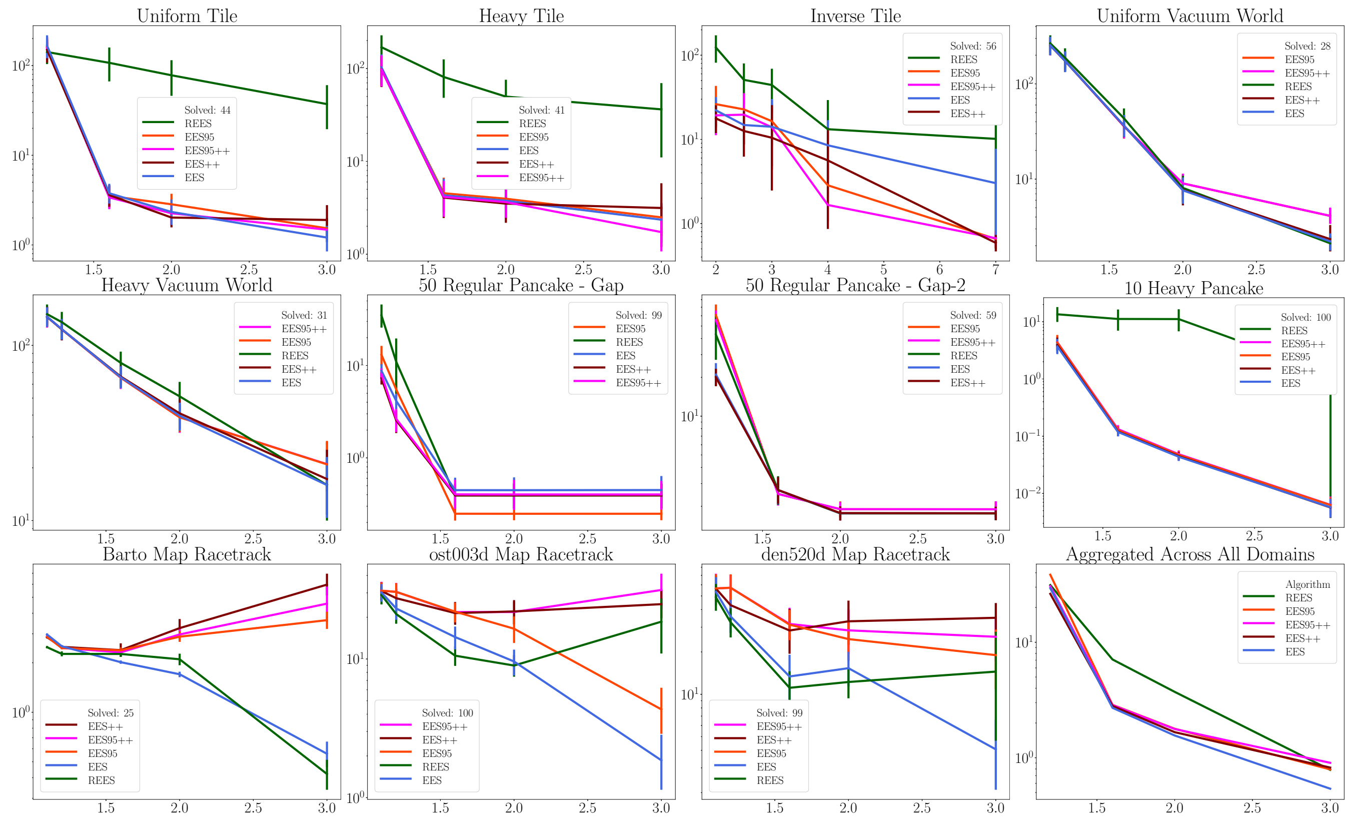

Figure 2: EES variants on search domains: CPU time (in seconds) as a function of the suboptimality bound.

the bounded-suboptimal setting as its adaptation DXES performs poorly here. A major reason is that DXES neglects to raise the suboptimality bound: when looking at its search behavior in comparison to algorithms such as EES, we found that f_{min} increases much more slowly throughout the search in DXES. This was the main motivation behind CDXES and the round-robin strategies, which explicitly aim to raise f_{min} by expanding $best_f$. Yet, as Figure 1 shows, CDXES performs even worse than DXES. By periodically expanding $best_f$, the probability estimates may significantly change due to fluctuations in f_{min} (and $\hat{f}_{min}$), leading to many nodes with outdated values in the open list. As an attempt to ameliorate this, we tested a variant of CDXES that lazily reevaluates nodes whose xe values change between insertion and expansion: the CDXES$\circlearrowleft$ configuration requeues nodes if their updated xe estimate differs by more than 5%. This significantly improves results, though it still does not reach the overall performance of WA* and DPS. We tested this approach also for DXES and RR-DXES, but found that the results of the former remain almost unchanged, but performance degrades for the latter. Similarly, a bucket-based reordering strategy akin to DPS also did not improve results for DXES.

Table 1 shows that CDXES$\circlearrowleft$ has merit in a few domains: it has strictly higher coverage than the other algorithms in Snake, and has equal (or better) coverage with fewer expansions in Grid, Miconic, Parcprinter, and TPP on commonly solved instances. Overall though, the XES variants for bounded-suboptimal do not carry over the success from bounded-cost search and are outperformed by the baselines in most domains. We also ran experiments with DXES on the search domains with similar results, i.e., performance was worse overall compared to the baselines. Due to its low coverage in many domains, we omit it in the figures for the search domains to retain a larger number of commonly solved instances. It appears that, even though it is possible to port bounded-cost algorithms to the bounded-suboptimal setting (Gilon, Felner, and Stern 2016), it is not trivial to achieve high performance.

Round-Robin Algorithms

Figure 3 compares the round-robin algorithms using d, ud, or xe in the focal list against the baselines weighted A*, EES, and DPS. In the sliding-tile domains, RR-d outperforms the state-of-the-art algorithms, and so does RR-DXES on the heavy- and inverse-cost versions (we omit weighted A*, DPS, and RR-DPS in inverse tiles due to their poor coverage). In the regular pancake domain, weighted A* and DPS show relatively strong performance, but they are again surpassed by the round-robin algorithms on the more difficult heavy-cost variant. Interestingly, the potential variant of the round-robin algorithms, RR-DPS, does not work as well as RR-d and RR-DXES, and is generally outperformed by its base version DPS. Aggregated over all search domains (see

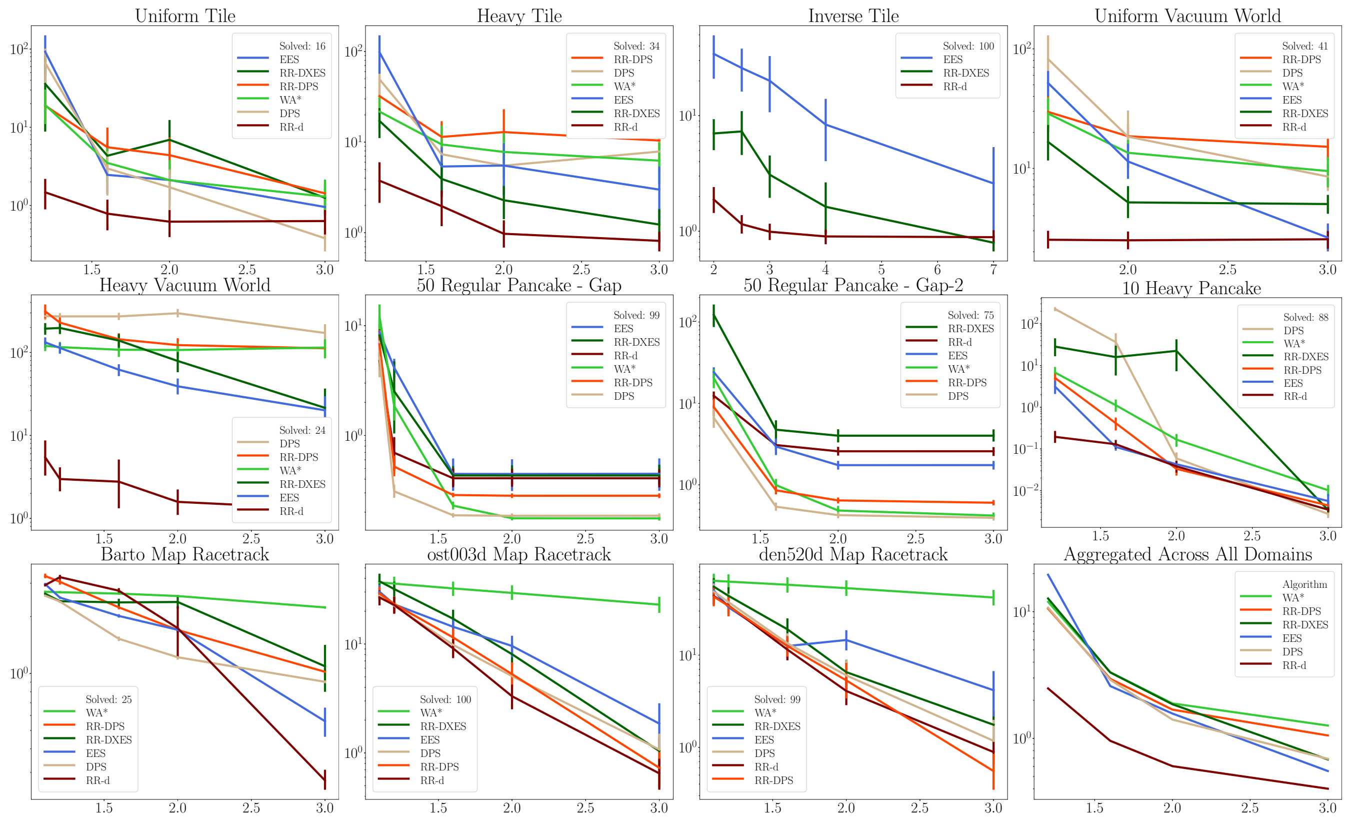

Figure 3: Round-robin variants on the search domains: CPU time (in seconds) as a function of the suboptimality bound.

the right bottom plot in Figure 3) RR-d performs the best overall, with a significant margin over the other algorithms, in particular for smaller suboptimality bounds.

The experiments on the planning domains paint a similar picture (consider again Figure 1 and Table 1). The round-robin variants RR-d and RR-DXES perform best overall, significantly outperforming the other algorithms across most tested suboptimality bounds. As observed on the search domains, RR-DPS loses out to its baseline DPS: for $w = 1.5$, RR-DPS improves coverage over DPS in 4 domains, but lowers it in 18. In contrast, RR-DXES demonstrates extremely robust performance. In terms of coverage, it is a strict improvement over DXES, boosting coverage in 38 (!) domains while never falling below DXES for $w = 1.5$; and almost halving the number of expansions on average. Excluding domains where all algorithms have the same coverage, RR-DXES is the (at least shared) best algorithm on 28 out of 42 domains. As the suboptimality bound increases, RR-d catches up and surpasses RR-DXES at $w = 4$ (omitted from Figure 1 for space).

In order to test whether periodically expanding both $best_f$ and $best_{\hat{f}}$ is important, we also tested configurations that alternate the focal queue only with either $best_f$ or $best_{\hat{f}}$, and found them to be inferior to the variants presented here. We conclude that $best_f$ and $best_{\hat{f}}$ are useful components for bounded-suboptimal search.

Conclusions

Our comprehensive empirical investigation considered several principled modifications to EES but, unfortunately, found that none of them improve performance significantly. We also considered several ways to exploit estimates of search effort, which was surprisingly difficult. The most effective approach was to eschew sophisticated metareasoning and merely use f, $\hat{f}$, and either d or xe within a simple round-robin queue framework.

Despite the state-of-the-art results obtained by RR-d and RR-DXES, the success of these round-robin algorithms suggests that plenty of room remains for developing principled selection mechanisms. It does not appear straightforward to estimate when a bounded-suboptimal algorithm should devote effort to improving the bound estimation versus pursuing promising solutions.

The strength of DPS in certain domains is also intriguing, as the algorithm does not explicitly optimize search speed. Merely incorporating potential into a round-robin framework does not seem sufficient to gain its benefits. Further research is necessary to understand how best to leverage probabilistic estimates in suboptimal search.

Acknowledgments

Many thanks to William Doyle for suggesting EES++ and to Jiaoyang Li for suggesting Rushed EES. We are grateful for support from DFG grant 389792660 as part of TRR 248

– CPEC (see https://perspicuous-computing.science) and from NSF-BSF via grant 2008594.

References

Allouche, D.; Barbe, S.; de Givry, S.; Katsirelos, G.; Lebbah, Y.; Loudni, S.; Ouali, A.; Schiex, T.; Simoncini, D.; and Zytnicki, M. 2019. Cost Function Networks to Solve Large Computational Protein Design Problems. In *Operations Research and Simulation in healthcare*. Springer.

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1995. Learning to act using real-time dynamic programming. *Artificial intelligence*, 72(1-2): 81–138.

Dionne, A. J.; Thayer, J. T.; and Ruml, W. 2011. Deadline-Aware Search Using On-Line Measures of Behavior. In *Proceedings of the Fourth Annual Symposium on Combinatorial Search, SOCS 2011*.

Ferguson, D.; Howard, T. M.; and Likhachev, M. 2008. Motion planning in urban environments. *Journal of Field Robotics*, 25(11-12): 939–960.

Fickert, M.; Gu, T.; and Ruml, W. 2021. Bounded-cost Search Using Estimates of Uncertainty. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 1675–1681.

Fickert, M.; Gu, T.; Staut, L.; Ruml, W.; Hoffmann, J.; and Petrik, M. 2020. Beliefs We Can Believe In: Replacing Assumptions with Data in Real-Time Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 9827–9834.

Gates, W. H.; and Papadimitriou, C. H. 1979. Bounds for sorting by prefix reversal. *Discrete mathematics*, 27(1): 47–57.

Gilon, D.; Felner, A.; and Stern, R. 2016. Dynamic potential search—a new bounded suboptimal search. In *Ninth Annual Symposium on Combinatorial Search*.

Gilon, D.; Felner, A.; and Stern, R. 2017. Dynamic Potential Search on Weighted Graphs. In *Proceedings of the Tenth Annual Symposium on Combinatorial Search*.

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

Helmert, M. 2010. Landmark heuristics for the pancake problem. In *Third Annual Symposium on Combinatorial Search*.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling*.

Helmert, M.; and Röger, G. 2008. How Good is Almost Perfect? In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, volume 8, 944–949.

Heydari, M. H.; and Sudborough, I. H. 1997. On the diameter of the pancake network. *Journal of Algorithms*, 25(1): 67–94.

Kim, K. H.; and Park, Y.-M. 2004. A crane scheduling method for port container terminals. *European Journal of Operational Research*, 156(3): 752–768.

Kleitman, D.; Kramer, E.; Conway, J.; Bell, S.; and Dweighter, H. 1975. Elementary Problems: E2564–E2569. *The American Mathematical Monthly*, 82(10): 1009–1010.

Korf, R. E. 1985. Iterative-Deepening-A*: An Optimal Admissible Tree Search. In *Proceedings of IJCAI-85*, 1034–1036.

Li, J.; Tinka, A.; Kiesel, S.; Durham, J. W.; Kumar, T. K. S.; and Koenig, S. 2021. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, 11272–11281.

Mitchell, A.; Ruml, W.; Spaniol, F.; Hoffmann, J.; and Petrik, M. 2019. Real-time planning as decision-making under uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2338–2345.

Mudgal, A.; Tovey, C.; Greenberg, S.; and Koenig, S. 2005. Bounds on the travel cost of a mars rover prototype search heuristic. *SIAM Journal on Discrete Mathematics*, 19(2): 431–447.

Pearl, J.; and Kim, J. H. 1982. Studies in Semi-Admissible Heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(4): 391–399.

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3-4): 193–204.

Röger, G.; and Helmert, M. 2010. The more, the merrier: Combining heuristic estimators for satisficing planning. In *Twentieth International Conference on Automated Planning and Scheduling*.

Seipp, J.; Pommerening, F.; Sievers, S.; and Helmert, M. 2017. Downward Lab. https://doi.org/10.5281/zenodo.790461.

Stern, R. T.; Puzis, R.; and Felner, A. 2011. Potential search: A bounded-cost search algorithm. In *Twenty-First International Conference on Automated Planning and Scheduling*.

Sturtevant, N. 2012. Benchmarks for Grid-Based Pathfinding. *Transactions on Computational Intelligence and AI in Games*, 4(2): 144–148.

Thayer, J. T.; Dionne, A.; and Ruml, W. 2011. Learning Inadmissible Heuristics During Search. In *Proceedings of the Twenty-first International Conference on Automated Planning and Scheduling (ICAPS-11)*.

Thayer, J. T.; and Ruml, W. 2011. Bounded suboptimal search: A direct approach using inadmissible estimates. In *IJCAI*, volume 2011, 674–679.

Thayer, J. T.; Ruml, W.; and Kreis, J. 2009. Using Distance Estimates in Heuristic Search: A Re-evaluation. In *Proceedings of the Symposium on Combinatorial Search (SoCS-09)*.

Thayer, J. T.; Stern, R.; Felner, A.; and Ruml, W. 2012. Faster bounded-cost search using inadmissible estimates. In *Twenty-Second International Conference on Automated Planning and Scheduling*.

Thiébaux, S.; Coffrin, C.; Hijazi, H.; and Slaney, J. 2013. Planning with MIP for supply restoration in power distribution systems. In *23rd International Joint Conference on Artificial Intelligence (IJCAI-13)*. AAAI Press.

Wilt, C. M.; and Ruml, W. 2014. Speedy Versus Greedy Search. In *Proceedings of the Seventh Annual Symposium on Combinatorial Search*.