Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)

The Two-Edged Nature of Diverse Action Costs

Gaojian Fan, Martin Müller, Robert Holte
Computing Science Department
University of Alberta
Edmonton, Canada T6G 2E8
{gaojian, mmueller, rholte}@ualberta.ca

Abstract

Diverse action costs are an essential feature of many real-world planning applications. Some recent studies have shown that diversity of action costs makes planning more difficult, and that searching using unit action costs can outperform searching the same domain with diverse action costs. In this paper, we provide experimental evidence and theoretical analysis showing that search can also benefit from action cost diversity. We show that on several IPC problems cost diversity has a positive effect (reduces search effort). We then present a theoretical analysis establishing that these positive cases are not accidental. Our main result is a "No Free Lunch" theorem showing that any negative effects of cost diversity are always perfectly counterbalanced by positive effects. Our theoretical analysis also shows that it is advantageous to have a strongly concentrated distribution of solution costs. In many domains, unit costs will give rise to a more concentrated distribution than diverse costs, but we give an example typifying domains in which the opposite is the case.

Introduction

Diverse action costs occur naturally in many planning problems. For example, in the IPC TRANSPORT domain, loading and unloading a package is much cheaper than moving a vehicle, and the cost of moving a vehicle varies widely with the distance between locations.

In recognition of its importance, planning with action costs has been studied in both the optimal and satisficing settings in recent years. One notable trend is a focus on the *negative* impact of diverse costs on planning. Several studies demonstrate cases where planning with diverse action costs is more difficult, in terms of the number of nodes expanded, than planning in the same domain with unit costs, where every action has a cost of 1 (Benton et al. 2010; Cushing, Benton, and Kambhampati 2011; Wilt and Ruml 2011; 2014). Other studies showed that some search algorithms perform better on domains with diverse action costs when they use information (e.g. heuristics) based on unit costs instead of relying entirely on information based on the given costs (Thayer and Ruml 2009; Richter and Westphal 2010; Thayer and Ruml 2011; Thayer, Benton, and Helmert 2012).

One clear disadvantage of diverse costs is a kind of "horizon effect". If a search space has huge regions reachable by low-cost actions, but the solution requires a high-cost action a, the low-cost regions will be exhaustively searched before the state s reached by a is expanded. In the unit cost model, state s would be expanded much earlier, allowing the solution to be found much more quickly.

Thus there is considerable evidence that *action cost diversity is harmful for search.* Is it true that this is generally the case? This is the question we address in this paper. We provide both theoretical and experimental investigations of the effects of changing action costs on the number of nodes expanded. Our experiments give a variety of examples, including problems from the International Planning Competition (IPC), where diversity of action costs is beneficial, i.e. the number of nodes expanded when the action costs are diverse is substantially lower than the number expanded using unit costs.

Our main theoretical result is a "No Free Lunch" (NFL) theorem about the impact, on the number of nodes expanded by Dijkstra's algorithm (Dijkstra 1959), of changing from any cost function $\mathcal{C}$ to any other $\mathcal{C}'$. We prove that for any given state space S, the problem instances in S on which fewer nodes are expanded using $\mathcal{C}$ are exactly counterbalanced by the problem instances on which fewer nodes are expanded using $\mathcal{C}'$. This implies, for example, that if start and goal states are selected randomly, the expected number of nodes expanded is the same no matter what cost function is used.

The NFL theorem applies to all state spaces, including the "ε-cost traps" that Cushing et al. (2010; 2011) designed to illustrate that search with non-unit action costs can expand exponentially more states than search with unit costs. For a particular type of trap, the cycle trap, we show experimentally that the total number of nodes expanded over all possible goal states is *almost* identical whether one uses unit costs or the alternative cost function defined by Cushing et al.

The reason the totals in the cycle trap experiment are not exactly equal is that the NFL theorem does not take into account states that are the same distance from the start state as the goal. We call such states TIE states. Our second theoretical contribution is to analyze the impact of TIE states. We show that unit costs will often have an advantage over

non-unit costs because of TIE states (as is the case for cycle traps) but we also describe a new planning domain—Hazardous Logistics—in which TIE states work to the advantage of non-unit costs.

The remainder of the paper is organized as follows. In Section we review previous work on the impact of diverse action costs on search. Section gives three motivating examples in which diverse action costs are beneficial. Section 4 describes the setting for our theoretical analysis and experimentally shows that in this setting many of the IPC problems from domains with non-unit costs are more easily solved using the non-unit costs than using unit costs. Section 5 contains the NFL theorem and the cycle trap experiments. Section 6 contains our theoretical study related to tie-breaking and the definition of the Hazardous Logistics domain.

Related Work

Some domains with non-unit action cost functions have been used in the International Planning Competition (IPC) since 2008. Many recent planners contain a search or evaluation component where the true action cost function $\mathcal{C}$ is replaced by the unit cost function $\mathcal{U}$. We will call such a component either $\mathcal{C}$-based or $\mathcal{U}$-based, depending on which cost function it uses. For example, a $\mathcal{U}$-based heuristic, $h(s)$, estimates the number of actions that must be applied to state s to reach a goal state.

The use of $\mathcal{U}$-based heuristics was empirically shown to improve the performance of many planners on IPC problems. For example, the first stage of LAMA (Richter, Westphal, and Helmert 2011; Richter and Westphal 2010), the winner of the IPC 2008 sequential satisficing track, uses $\mathcal{U}$-based FF and landmark heuristics. Later stages use $\mathcal{C}$-based heuristics. Explicit Estimation Search (Thayer and Ruml 2011) uses both $\mathcal{C}$-based and $\mathcal{U}$-based heuristics to determine the next node to expand in bounded suboptimal search. Wilt and Ruml (2011) give examples where $\mathcal{U}$-based search algorithms outperform $\mathcal{C}$-based ones. They also demonstrate cases where $\mathcal{U}$-based GBFS expands fewer nodes than $\mathcal{C}$-based GBFS (Wilt and Ruml 2014). The A* algorithm sometimes performs better with $\mathcal{U}$ than with $\mathcal{C}$ for the state spaces studied in (Wilt and Ruml 2011; Wilt 2014).

Along with these empirical observations, the following explanations for why $\mathcal{U}$-based search is easier than $\mathcal{C}$-based search were proposed. Wilt and Ruml (2011; 2014) argue that the difficulty of $\mathcal{C}$-based search is due to large heuristic errors and large local minima caused by action costs with high variance. Cushing, Benton and Kambhampati (2011) give a theoretical analysis of Dijkstra's algorithm and provide examples where $\mathcal{C}$-based search expands exponentially many more nodes than $\mathcal{U}$-based search.

All these studies seem to indicate that *diverse action costs are harmful for best-first search* in planning. While there is evidence to support this belief, it is not the full story. The parameterized complexity analysis by Aghighi and Bäckström (2015; 2016) show that optimal planning is no harder with positive integer costs than with unit costs, and that the difference between maximal and minimal costs is irrelevant to the inherent complexity of planning. The literature also contains some evidence that diverse action costs can speed up search. In Wilt's study of the effect of costs on A* (Wilt 2014), in two out of the five domains, A* expands fewer nodes for higher variance action costs. In two of four domains used by Wilt and Ruml (2014), GBFS expands fewer nodes with non-unit action costs. In the IPC domains FLOORTILE and WOODWORKING, GBFS solves more problems when its FF heuristic is $\mathcal{C}$-based than when it is $\mathcal{U}$-based (Nakhost 2013).

Motivating Examples

The following examples give evidence that diverse action costs can be beneficial for search. Example 1 also foreshadows the use of surely expanded nodes in our analysis in Section . Example 2 illustrates the general gist of our later NFL theorem by showing that for a previously published example in which diverse costs are harmful, there exist other examples with the opposite search behaviour.

Example 1: IPC-11 PARCPRINTER Problem p08

To illustrate that diverse costs $\mathcal{C}$ can result in faster search than unit cost $\mathcal{U}$, consider the behaviour of A* with Fast-Downward's pattern database heuristic[1] on problem p08 from the IPC-2011 PARCPRINTER domain. The original IPC cost function $\mathcal{C}$ assigns a variety of costs to actions, ranging from 125 to 127,790. A* using $\mathcal{C}$ expands 61,458 states to solve this problem (Figure 1), while using $\mathcal{U}$ it requires 2,453,418 node expansions (Figure 2), more than 39 times as many.

The additional state expansions are not due to unlucky tie-breaking with $\mathcal{U}$. With a consistent heuristic, A* must expand every state s for which $f(s) < C^*$, the optimal solution cost (Pearl 1984). Such states are called "surely expanded", or SE for short. In this problem instance, there are 61,456 SE states under cost function $\mathcal{C}$ —all states to the left of the red vertical line in Figure 1—but 2,453,415 under $\mathcal{U}$, shown to the left of the red vertical line in Figure 2. When using $\mathcal{U}$, A* must expand millions of states more than with $\mathcal{C}$.

Example 2: 15-Puzzle

Consider the following successively more diverse cost functions for the 15-puzzle: unit cost $\mathcal{U}$, "linear" cost $\mathcal{L}$, and "square" cost $\mathcal{L}^2$, where an action a that moves tile T has $\mathcal{L}(a) = T$ and $\mathcal{L}^2(a) = T^2$. Wilt and Ruml (2011) show a 15-puzzle instance for which the number of nodes expanded by A*, with a suitably adjusted definition of Manhattan Distance (MD), increases dramatically when changing from $\mathcal{U}$ to $\mathcal{L}$ and from $\mathcal{L}$ to $\mathcal{L}^2$. This is illustrated in column $3(a)$ of Table 1, which shows the number of nodes expanded using each of these cost functions, for the instance with initial state shown in Figure 3(a) and the standard goal state of the 15-puzzle. When using $\mathcal{L}^2$, A* ran out of memory with a 2GB limit on this problem.

In their example, tiles 14 and 15 occupy each other's goal locations in the initial state so each of these tiles must be

[1] In its default setting (www.fast-downward.org/Doc/Heuristic#Pattern_database_heuristic)

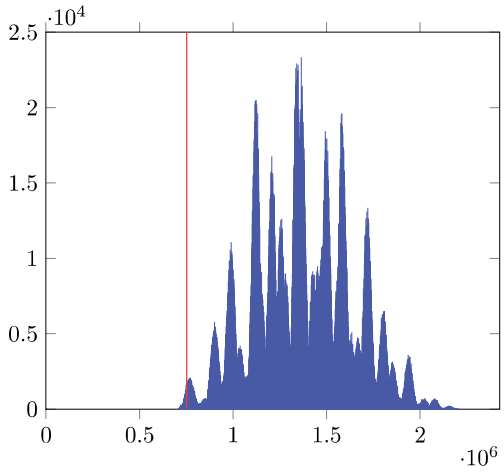

Figure 1: Histogram of f-values and optimal solution cost (751,642, the red vertical line) for IPC-2011 PARCPRINTER problem p08 with cost function $\mathcal{C}$.

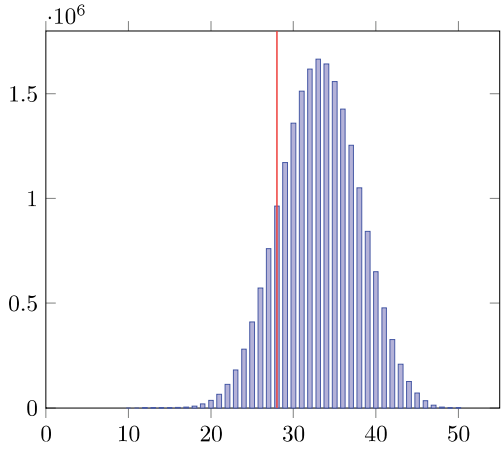

Figure 2: Histogram for f-values and optimal solution cost (28, the red vertical line) for IPC-2011 PARCPRINTER problem p08 with cost function $\mathcal{U}$.

moved at least once, which requires using the most expensive actions under $\mathcal{L}$ and $\mathcal{L}^2$. For problems whose solution path does not necessarily use the expensive actions, we observe the opposite trend. For example, "reversing" the tile numbers (i.e., each tile T is replaced with $16 - T$) maps the state in Figure 3(a) to the state in Figure 3(b) and maps the standard goal state to the state in Figure 3(c). The number of nodes expanded by A* to solve this problem is shown in the "Reverse" column of Table 1 and here we see the opposite trend: the number decreases as the cost diversity increases.

Example 3: Heuristics = Diverse Action Costs

As our final motivating example, the very idea of using a heuristic to guide search can be viewed as taking advantage of action cost diversity, in the following sense. It is well known (Martelli 1977; Ikeda et al. 1994; Edelkamp and

	$3(a)$	Reverse
$\mathcal{U}$	33,798	33,798
$\mathcal{L}$	1,841,122	3,660
$\mathcal{L}^2$	not solved	1,274

Table 1: Number of nodes expanded by A* for each cost function on two different problems.

	1	2	3
4	5	6	7
8	9	10	11
13	**12**	**15**	**14**

(a)

	15	14	13
12	11	10	9
8	7	6	5
3	**4**	**1**	**2**

(b)

	15	14	13
12	11	10	9
8	7	6	5
4	3	2	1

(c)

Figure 3: (a) tiles 14 and 15 in each other's goal positions; (b) state (a) "reversed"; (c) standard goal state "reversed".

Schrödl 2012) that A* using a consistent heuristic h in combination with the original action costs behaves the same as Dijkstra's algorithm using no heuristic but with action costs adjusted as follows. If $c(s, a, t)$ is the original cost of the action a leading from state s to state t then the adjusted cost is $c'(s, a, t) = c(s, a, t) - h(s) + h(t)$. In this view, the guidance a heuristic provides towards the goal is converted to *guidance towards the goal by diversified cost*: the cost of actions moving towards the goal (as estimated by the heuristic) is reduced, while the cost of actions leading away from the goal is increased. As an illustration, consider the Sliding Tile Puzzle with unit cost and the Manhattan Distance heuristic (MD). Any action that moves a tile towards its goal position decreases MD by 1, so the adjusted cost of this action is 0. In contrast, any action that moves a tile away from its goal position increases MD by 1 and has an adjusted cost of 2. A* solves problems much faster with MD than with no heuristic, and Dijkstra's algorithm with the corresponding adjusted action costs (but no heuristic) will see the same speedup compared to searching with unit costs. The adjusted cost function has costs of 0 and 2 and is therefore more diverse than unit costs.

Theoretical Setting

The theoretical analysis that follows is described in terms of forward search, i.e. search beginning at the start state and applying the actions in the usual way to compute the successors of a state. Because we assume that a problem instance specifies a start state and a goal state (not a goal condition), the theory applies, with no changes whatsoever, to backward search. Our theory is not currently capable of taking into account the effects of a heuristic function, so its application is restricted to Dijkstra's shortest path algorithm, which is equivalent to A* with no heuristic.

To see whether diverse costs can be beneficial when search is being done without a heuristic, we have run Dijkstra's algorithm on all 12 of the IPC domains that have a non-unit action cost function $\mathcal{C}$. All these domains use integer costs. 0 action cost appears in five domains among which

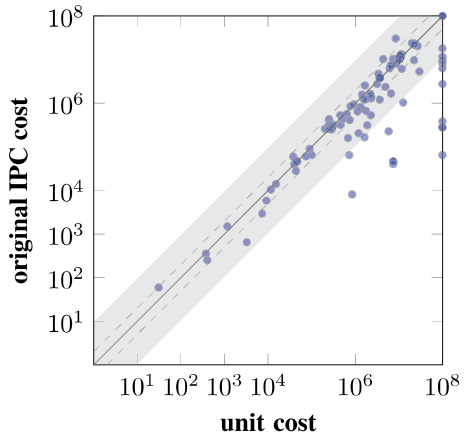

Figure 4: Number of nodes $N_{\mathcal{C}}$ expanded using the original IPC cost function (y-axis) vs. $N_{\mathcal{U}}$, nodes expanded using the unit cost function (x-axis).

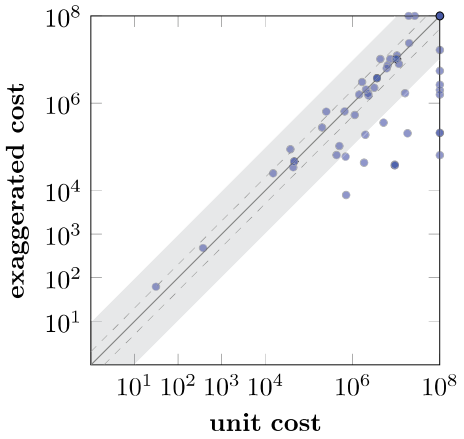

Figure 5: Number of nodes expanded using an exaggerated cost function (y-axis) vs. the number expanded using the unit cost function (x-axis).

three domains (SOKOBAN, PEGSOL and OPENSTACKS) have only 0 and 1 as their action costs.

Figure 4 is a log-log plot comparing $N_{\mathcal{C}}$ on the y-axis with $N_{\mathcal{U}}$ on the x-axis for each IPC instance in these domains, where $N_{\mathcal{C}}$ is the number of nodes expanded to solve the instance using $\mathcal{C}$, and $N_{\mathcal{U}}$ is the corresponding number for the unit cost function $\mathcal{U}$. Points below the $y = x$ line represent instances that required more nodes to be expanded using $\mathcal{U}$ than using $\mathcal{C}$ (i.e. $N_{\mathcal{U}} > N_{\mathcal{C}}$). The dashed lines just above and below the $y = x$ line are $y = 2x$ and $y = \frac{x}{2}$ respectively. For points below the shaded zone, $N_{\mathcal{U}} > 10N_{\mathcal{C}}$, so solving them with $\mathcal{U}$ requires more than $10\times$ as many nodes as with $\mathcal{C}$. Points on the right-most (resp., top-most) edge of the plot are instances that were not solved using $\mathcal{U}$ (resp., $\mathcal{C}$) within the 2G memory limit.

Overall, more nodes are expanded for $\mathcal{U}$ than for $\mathcal{C}$ to solve these instances. On no instance is $N_{\mathcal{C}} > 3.6N_{\mathcal{U}}$ and only for 2 instances is $N_{\mathcal{C}} > 2N_{\mathcal{U}}$. By contrast, there are 30 instances for which $N_{\mathcal{U}} > 2N_{\mathcal{C}}$, and 10 instances are solved with $\mathcal{C}$ but not with $\mathcal{U}$.

In some of these domains, even though $\mathcal{C}$ is non-unit, the ratio of the maximum cost to the minimum cost in $\mathcal{C}$ is not large. On those domains we re-ran the experiment with an exaggerated cost function $\mathcal{E}$ in which the i-th largest cost in $\mathcal{C}$ is replaced by 10^i if it is smaller than 10^i.[2] Figure 5 shows the results, with $N_{\mathcal{E}}$ on the y-axis and $N_{\mathcal{U}}$ on the x-axis. Overall, Figure 5 is very similar to Figure 4, showing that the trends seen in Figure 4 are not caused by some peculiar property of $\mathcal{C}$ in these domains.

Formal Preliminaries

We choose a formulation which clearly separates the two main components of a planning problem, as in Katz & Domshlak (2008):

[2]For the TRANSPORT domain only, we used 2^i instead of 10^i to avoid numerical overflow problems in FastDownward.

- A *transition graph structure (TGS)* is a triple $\mathcal{S} = \langle S, L, T \rangle$ where the state space S is a finite set of states, L is a finite set of transition labels, and $T \subseteq S \times L \times S$ is a set of labelled transitions.
- A *cost function* for L is a mapping $\mathcal{C} : L \mapsto \mathbb{R}_0^+$, where $\mathbb{R}_0^+$ is the set of non-negative real numbers.

A *transition graph* is a pair $\mathcal{T} = \langle \mathcal{S}, \mathcal{C} \rangle$, where $\mathcal{S}$ is a transition graph structure with label set L, and $\mathcal{C}$ is a cost function for L.

A *planning problem* in a transition graph structure $\mathcal{S} = \langle S, L, T \rangle$ is a pair $\langle s_{\text{init}}, goal \rangle$ where $s_{\text{init}} \in S$ is the initial state, and $goal \in S$ is the goal state.

A *path* in $\mathcal{S}$ is a sequence $(s_0, l_1, s_1, ..., s_{n-1}, l_n, s_n)$ such that $n \in \mathbb{N}_0$ and $(s_{i-1}, l_i, s_i) \in T$ for $i \in \{1, 2, ..., n\}$. In our analysis we are only concerned with paths having $s_0 = s_{\text{init}}$, so we henceforth assume this holds. Path P is a *solution* for problem $\langle s_{\text{init}}, goal \rangle$ if its final state is $goal$. The cost of path $P = (s_0, l_1, s_1, ..., s_{n-1}, l_n, s_n)$ is $\sum_{j=1}^{n} \mathcal{C}(l_j)$.

$d_{\mathcal{C}}(s, t)$ is the distance (cost of a least-cost path) from state s to state t when the cost function is $\mathcal{C}$. We abbreviate $d_{\mathcal{C}}(s_{\text{init}}, t)$ as $d_{\mathcal{C}}(t)$. The optimal solution cost is $d_{\mathcal{C}}(goal)$.

SE denotes the set of states that are "surely expanded" by Dijkstra's algorithm, which are the states that are strictly closer to s_{init} than $goal$, i.e. $\{s \mid d_{\mathcal{C}}(s) < d_{\mathcal{C}}(goal)\}$. NE ("never expanded") denotes the set of states strictly further from s_{init} than $goal$, i.e. $\{s \mid d_{\mathcal{C}}(s) > d_{\mathcal{C}}(goal)\}$. Under no conditions will Dijkstra's algorithm expand states in NE. TIE is the set of states s with $d_{\mathcal{C}}(s) = d_{\mathcal{C}}(goal)$. Dijkstra's algorithm might expand some (or all) of the TIE states, or it might expand none.

These sets are, of course, functions of all the variables that define a planning problem—$\mathcal{S}$, $\mathcal{C}$, s_{init}, and $goal$—but in the following analysis $\mathcal{S}$ and s_{init} are held fixed, so we will write SE($\mathcal{C}$,$goal$), and similarly for NE and TIE.

We begin our analysis by ignoring TIE in Section . Section then extends the analysis to account for these states.

No Free Lunch Theorem

In this section, for a fixed transition graph structure $\mathcal{S}$ and start state s_{init}, we compare the SE and NE sets for two different cost functions, $\mathcal{C}$ and $\mathcal{C}'$, considering all possible goal states. Definition 1 states that the pair of states (s,t) *favors* $\mathcal{C}$ *over* $\mathcal{C}'$ if, when the goal is t, s is surely expanded when the cost function is $\mathcal{C}'$ but never expanded when it is $\mathcal{C}$. Theorem 2 states that the number of pairs that favor $\mathcal{C}$ over $\mathcal{C}'$ is exactly the same as the number of pairs that favor $\mathcal{C}'$ over $\mathcal{C}$. This immediately implies Theorem 3, the "No Free Lunch" (NFL) theorem, which can be paraphrased thus: for a given start state, no cost function is any "easier" for Dijkstra's algorithm than any other cost function, when all possible goal states are taken into consideration and TIE states are ignored.

We consider all possible goal states but fix the initial state because the key to our NFL theorem is to take an average of the performance difference between $\mathcal{C}$ and $\mathcal{C}'$ over a set of problem instances, and the minimal set we could find that had an average of 0 was the set defined by any one initial state paired with all possible goal states. This zero average immediately implies a zero average over any set of initial states (for example, all possible initial states), as long as each is paired with all possible goal states.

The key fact underpinning our analysis is the following simple observation:

Lemma 1. *For any cost function $\mathcal{C}$ and any states s and t, $s \in SE(\mathcal{C},t)$ if and only if $t \in NE(\mathcal{C},s)$.*

Proof.

$$\begin{aligned} s \in SE(\mathcal{C},t) &\iff d_{\mathcal{C}}(s) < d_{\mathcal{C}}(t) \\ &\iff d_{\mathcal{C}}(t) > d_{\mathcal{C}}(s) \\ &\iff t \in NE(\mathcal{C},s)\,. \end{aligned}$$

□

Definition 1. *A pair of states (s,t) favors $\mathcal{C}$ over $\mathcal{C}'$ if $s \in NE(\mathcal{C},t)$ and $s \in SE(\mathcal{C}',t)$. $favor(\mathcal{C},\mathcal{C}')$ is the set of state pairs that favor $\mathcal{C}$ over $\mathcal{C}'$.*

Applying Lemma 1 to the definitions of $favor(\mathcal{C},\mathcal{C}')$ and $favor(\mathcal{C}',\mathcal{C})$ gives the following:

Theorem 2. *For any cost functions $\mathcal{C}$ and $\mathcal{C}'$ there is a one-to-one correspondence between the elements of $favor(\mathcal{C},\mathcal{C}')$ and $favor(\mathcal{C}',\mathcal{C})$.*

Proof. For any $(s,t) \in favor(\mathcal{C},\mathcal{C}')$, $s \in \text{NE}(\mathcal{C},t)$ and $s \in \text{SE}(\mathcal{C}',t)$. This implies, by Lemma 1, that $t \in \text{SE}(\mathcal{C},s)$ and $t \in \text{NE}(\mathcal{C}',s)$, i.e., that $(t,s) \in favor(\mathcal{C}',\mathcal{C})$. Therefore the mapping $\phi(s,t) = (t,s)$ is a one-to-one correspondence between $favor(\mathcal{C},\mathcal{C}')$ and $favor(\mathcal{C}',\mathcal{C})$. □

Definition 2. *For cost functions $\mathcal{C}$ and $\mathcal{C}'$ and state t, $\delta_t(\mathcal{C},\mathcal{C}')$ is the number of states s such that (s,t) favors $\mathcal{C}$ over $\mathcal{C}'$.*

By definition, we have

$$\begin{aligned} |\,favor(\mathcal{C},\mathcal{C}')| &= \Big|\bigcup_{t\in S}\{(s,t) \mid (s,t) \in favor(\mathcal{C},\mathcal{C}')\}\Big| \\ &= \sum_{t\in S}\delta_t(\mathcal{C},\mathcal{C}')\,. \end{aligned} \tag{1}$$

Let $\Delta_t(\mathcal{C},\mathcal{C}') = \delta_t(\mathcal{C},\mathcal{C}') - \delta_t(\mathcal{C}',\mathcal{C})$ whose absolute value $|\Delta_t(\mathcal{C},\mathcal{C}')|$ indicates how many *more* (if $\Delta_t(\mathcal{C},\mathcal{C}') > 0$) or *fewer* (if $\Delta_t(\mathcal{C},\mathcal{C}') < 0$) states are expanded for solving the problem when the goal is t with $\mathcal{C}'$ than with $\mathcal{C}$. By definition, $\Delta_t(\mathcal{C},\mathcal{C}') = -\Delta_t(\mathcal{C}',\mathcal{C})$.

Definition 3. *For cost functions $\mathcal{C}$ and $\mathcal{C}'$ we say state t is a goal state that favors $\mathcal{C}$ over $\mathcal{C}'$ if $\Delta_t(\mathcal{C},\mathcal{C}') > 0$. goals-favor$(\mathcal{C},\mathcal{C}')$ is the set of goal states that favor $\mathcal{C}$ over $\mathcal{C}'$.*

The following theorem shows that the advantage that $\mathcal{C}$ enjoys over $\mathcal{C}'$ on the goal states that favor $\mathcal{C}$ over $\mathcal{C}'$ is exactly counterbalanced by the disadvantage it suffers on the goal states that favors $\mathcal{C}'$ over $\mathcal{C}$.

Theorem 3 (No Free Lunch Theorem)**.** *For any two cost functions $\mathcal{C}$ and $\mathcal{C}'$,*

$$\sum_{t\in \textit{goals-favor}(\mathcal{C},\mathcal{C}')} \Delta_t(\mathcal{C},\mathcal{C}') \;=\; \sum_{t\in \textit{goals-favor}(\mathcal{C}',\mathcal{C})} \Delta_t(\mathcal{C}',\mathcal{C})\,.$$

Proof.

$$\begin{aligned} & |\,favor(\mathcal{C},\mathcal{C}')| = |\,favor(\mathcal{C}',\mathcal{C})| && \text{(by Theorem 2)} \\ \iff & \sum_{t\in S}\delta_t(\mathcal{C},\mathcal{C}') = \sum_{t\in S}\delta_t(\mathcal{C}',\mathcal{C}) && \text{(by Equation (1))} \\ \iff & 0 = \sum_{t\in S}\delta_t(\mathcal{C},\mathcal{C}') - \delta_t(\mathcal{C}',\mathcal{C}) = \sum_{t\in S}\Delta_t(\mathcal{C},\mathcal{C}')\,. && (2) \end{aligned}$$

Let $T^+ = \textit{goals-favor}(\mathcal{C},\mathcal{C}') = \{t \mid \Delta_t(\mathcal{C},\mathcal{C}') > 0\}$, $T^- = \textit{goals-favor}(\mathcal{C}',\mathcal{C}) = \{t \mid \Delta_t(\mathcal{C},\mathcal{C}') < 0\}$ and $T^0 = \{t \mid \Delta_t(\mathcal{C},\mathcal{C}') = 0\}$. Since $S = T^+ \cup T^- \cup T^0$, we have

$$\begin{aligned} & \text{Equation (2)} \\ \iff & 0 \;=\; \sum_{t\in T^+\cup T^-\cup T^0} \Delta_t(\mathcal{C},\mathcal{C}') \;=\; \sum_{t\in T^+\cup T^-} \Delta_t(\mathcal{C},\mathcal{C}') \\ \iff & 0 \;=\; \sum_{t\in T^+} \Delta_t(\mathcal{C},\mathcal{C}') + \sum_{t\in T^-} \Delta_t(\mathcal{C},\mathcal{C}') \\ \iff & 0 \;=\; \sum_{t\in \textit{goals-favor}(\mathcal{C},\mathcal{C}')} \Delta_t(\mathcal{C},\mathcal{C}') \;+\; \sum_{t\in \textit{goals-favor}(\mathcal{C}',\mathcal{C})} \Delta_t(\mathcal{C},\mathcal{C}') \\ \iff & 0 \;=\; \sum_{t\in \textit{goals-favor}(\mathcal{C},\mathcal{C}')} \Delta_t(\mathcal{C},\mathcal{C}') \;-\; \sum_{t\in \textit{goals-favor}(\mathcal{C}',\mathcal{C})} \Delta_t(\mathcal{C}',\mathcal{C}) \\ \iff & \sum_{t\in \textit{goals-favor}(\mathcal{C},\mathcal{C}')} \Delta_t(\mathcal{C},\mathcal{C}') \;=\; \sum_{t\in \textit{goals-favor}(\mathcal{C}',\mathcal{C})} \Delta_t(\mathcal{C}',\mathcal{C})\,. \end{aligned}$$

□

From a probabilistic point of view, Theorem 2 also implies the expected performance difference between two cost functions is zero when goal states are drawn uniformly at random and TIE states are ignored.

Theorem 4. *Let $\boldsymbol{D}$ be the random variable for $\Delta_t(\mathcal{C}, \mathcal{C}')$ when the goal state t is drawn uniformly at random from the state space S. The expected value of $\boldsymbol{D}$ is zero, i.e.,*

$$\mathbf{E}[\boldsymbol{D}] = \sum_d d \cdot P(\boldsymbol{D} = d) = 0$$

where $P(\boldsymbol{D} = d) = \dfrac{|\{t \mid \Delta_t(\mathcal{C}, \mathcal{C}') = d\}|}{|S|}$.

Proof. Let $T_d = \{t \mid \Delta_t(\mathcal{C}, \mathcal{C}') = d\}$. For any d,

$$\frac{\sum_{t \in T_d} \Delta_t(\mathcal{C}, \mathcal{C}')}{|S|} = d \cdot P(\boldsymbol{D} = d)\,.$$

As $\sum_{t \in S} \Delta_t(\mathcal{C}, \mathcal{C}') = 0$ (Equation (2)) and $S = \bigcup_d T_d$,

$$\sum_d d \cdot P(\boldsymbol{D} = d) = \frac{\sum_{t \in S} \Delta_t(\mathcal{C}, \mathcal{C}')}{|S|} = 0\,.$$

□

Example: ε-Cost Cycle Traps

The No Free Lunch (NFL) theorem applies to all state spaces, including the "ε-cost traps" that Cushing et al. (2010; 2011) designed to illustrate that search with a non-unit cost function $\mathcal{C}$ can expand exponentially more states than search with the unit cost function $\mathcal{U}$. In their discussion of these traps, Cushing et al. focused on the specific goal states that revealed this exponential difference. Our NFL theorem says that these differences in favor of $\mathcal{U}$ will be exactly balanced by performance on other goal states in favor of $\mathcal{C}$.

We will illustrate this with one of the traps Cushing et al. defined, the cycle trap. The states in a cycle trap are the integers from 0 to $2^k - 1$ for some k. Actions increment or decrement an integer modulo 2^k, including the overflow increment (increase $2^k - 1$ to 0), and the overflow decrement (decrease 0 to $2^k - 1$). In the non-unit cost function $\mathcal{C}$ defined by Cushing et al. the normal increment/decrement actions cost 1 and the overflow actions cost 2^{k-1}. Figure 6 shows the cycle trap for $k = 3$.

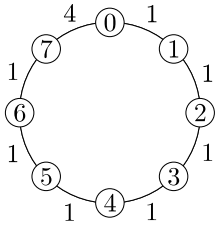

Figure 6: The cycle trap for $k = 3$. Numbers in circles are state IDs. Numbers next to edges are action costs.

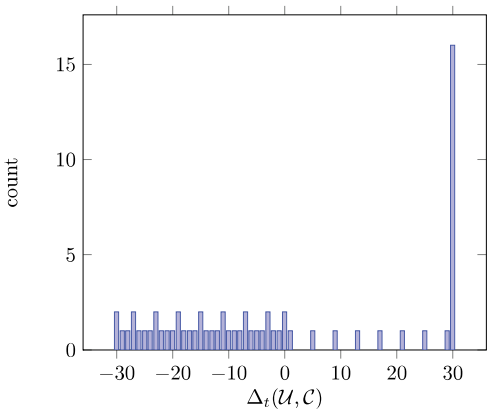

Figure 7: Histogram for $\Delta_t(\mathcal{U}, \mathcal{C})$, for the cycle trap with $k = 6$.

For initial state $s_{\text{init}} = 0$, if the goal state t is close to $2^k - 1$ then many fewer nodes will be expanded using $\mathcal{U}$ than using $\mathcal{C}$. For example, when $t = 2^k - 2$, Dijkstra's algorithm using $\mathcal{U}$ expands 3 states while the search using $\mathcal{C}$ has to expand $2^{k-1} + 2$ states. The NFL theorem says that this advantage for $\mathcal{U}$ on goal states such as $2^k - 2$ is exactly counterbalanced by disadvantages for $\mathcal{U}$ on other goal states. Figure 7 illustrates this for the cycle trap for $k = 6$. It shows how many states t have a given value of $\Delta_t(\mathcal{U}, \mathcal{C})$ (x-axis). There are 64 states in this space, and a quarter of them have a large positive $\Delta_t(\mathcal{U}, \mathcal{C})$ value (the spike at the right edge of the figure). However, these and all other states with $\Delta_t(\mathcal{U}, \mathcal{C}) > 0$ are exactly counterbalanced by the states with $\Delta_t(\mathcal{U}, \mathcal{C}) < 0$, i.e., their sum is equal to zero.

Table 2 shows $N_{\mathcal{C}}$ and $N_{\mathcal{U}}$, the total number of nodes expanded by Dijkstra's algorithm with all possible goal states

k	$N_{\mathcal{U}}(k)$	$N_{\mathcal{C}}(k)$
2	5	6
3	25	26
4	113	116
5	481	488
6	1985	2000
7	8065	8096
8	32513	32576
9	130561	130688
10	523265	523520
11	2095105	2095616
12	8384513	8385536
13	33546241	33548288
14	134201345	134205440
15	536838145	536846336
16	2147418113	2147434496

Table 2: Total number of nodes expanded with cost functions $\mathcal{U}$ and $\mathcal{C}$ for cycle traps of various sizes.

on cycle traps of sizes $k = 2$ to $k = 16$ using $\mathcal{C}$ and $\mathcal{U}$ respectively.

While the values are very close, $N_{\mathcal{C}}$ is always slightly larger than $N_{\mathcal{U}}$. This can also be seen in the following formulas, which give the exact number of nodes expanded for any k:

$$N_{\mathcal{U}}(k) = 2^{2k-1} - 2^k + 1\,,$$
$$N_{\mathcal{C}}(k) = 2^{2k-1} - 3 \cdot 2^{k-2}\,,$$
$$N_{\mathcal{C}}(k) - N_{\mathcal{U}}(k) = 2^{k-2} - 1.$$

The difference $N_{\mathcal{C}}(k) - N_{\mathcal{U}}(k)$ is tiny relative to the total number of states expanded, but not zero, because all cycle trap state spaces contain slightly more TIE states with $\mathcal{U}$ than they do with $\mathcal{C}$. The NFL theorem guarantees that $N_{\mathcal{C}}$ and $N_{\mathcal{U}}$ would be exactly equal if the effect of the TIE states was removed. In Section we show that TIE states often (but not always) work in favor of the unit cost distribution, as they did in this example.

Goal-Preference Tie-Breaking

The NFL theorem holds as long as there is no bias[3] to a particular problem taken into consideration. Such problem-specific biases include the tie-breaking strategy of best-first search, which breaks ties in favor of a goal state of a particular problem: goal states are expanded before non-goal states. When this tie-breaking strategy is used, the expansion order of TIE states can be different for different goals, and thus the one-to-one correspondence in Theorem 2 is not guaranteed to exist between those states.

For any cost function, let $n \in \mathbb{N}$ be the number of distances from the initial state to all states in the state space, let A_i for $i \in \{1, 2, ..., n\}$ be the set of states that have the i-th smallest distance, and let $a_i = |A_i|$ for $i \in \{1, 2, ..., n\}$. If the goal state is in A_i, then for all $j < i$ the states in A_j are surely expanded regardless of the tie-breaking. The total number of surely expanded states for all goal states in A_i is $SE(A_i) = a_i \sum_{j=1}^{i-1} a_j$. States in A_i are the TIE states whose expansions depend on the tie-breaking strategy. Let a *bias-free* search be Dijkstra's search algorithm that does not break ties in favor of a goal state and expands TIE states in a fixed order. For a bias-free search, the total number of states expanded when the goal state is from A_i is $SE(A_i) + 0 + 1 + 2 + \cdots + (a_i - 1) = SE(A_i) + \frac{a_i \cdot (a_i - 1)}{2}$, where $\frac{a_i \cdot (a_i - 1)}{2}$ is the cumulative number of extra expansions in addition to the sure expansions for all goal states in A_i by the bias-free search. If Dijkstra's algorithm breaks ties in favor of a goal state and there are no zero cost actions, none of these extra expansions by the bias-free search are needed before the goal from A_i is found. In this best case, the total number of expansions *saved* by tie-breaking over all goal states from all A_i is

$$\sum_{i=1}^{n} \frac{a_i \cdot (a_i - 1)}{2} = \frac{1}{2} \sum_{i=1}^{n} a_i^2 - \frac{|S|}{2}\,, \tag{3}$$

[3]Excluding problem bias is also required for NFL theorems for general search and optimization (Wolpert and Macready 1995; 1997).

	unit	linear	square	inverse	sqrt
Predict	82,676	88,966	90,426	90,501	90,663
Actual	82,185	88,318	89,714	90,026	90,021

Table 3: The actual average performance of the search and the prediction based on Equation (4) on 8-puzzle.

and the total number of expansions by Dijkstra's algorithm is

$$\sum_{i=1}^{n} SE(A_i) = \sum_{i=1}^{n} a_i \cdot \sum_{j=1}^{i-1} a_j$$
$$= \frac{(a_1 + a_2 + \cdots + a_n)^2 - (a_1^2 + a_2^2 + \cdots + a_n^2)}{2}$$
$$= \frac{1}{2}(|S|^2 - \sum_{i=1}^{n} a_i^2) \tag{4}$$

The savings due to the best-case tie-breaking depend on the sum-of-squares $\sum_{i=1}^{n} a_i^2$. By Equation (3), the larger this sum is, the more expansions are saved by tie-breaking, and by Equation (4), the fewer states are expanded by Dijkstra's algorithm with tie-breaking. This sum of squares is smaller when the distribution of a_i is more spread out and the minimum value is achieved when $a_i = 1$ for all i. The sum is larger when the distribution is more concentrated, and its maximum is achieved for $n = 1$ and $a_1 = |S|$. Although the sum-of-squares could be as large as $|S|^2$, this extreme case is only achieved when all action costs are 0 (and the best-case tie-breaking is not guaranteed). In problems encountered in practice, the sum is much smaller than $|S|^2$.

Our experiments on the 8-puzzle show that the actual average performance of the search matches the prediction of Equation (4). We tested the five cost functions for the 8-puzzle used previously (Wilt and Ruml 2011; Wilt 2014). In Table 3, the entry "Predict" contains the average of $(|S|^2 - \sum_{i=1}^{n} a_i^2)/2$, over 1,500 random initial states from which the whole distributions are generated. The "Actual" entry shows the average performance of search over 10,000 random pairs of initial and goal states.

The benefit of the tie-breaking is linked to how concentrated the distance distribution is. On one hand, cost functions that provide diverse action costs but no 0 cost seem to naturally induce a more spread out distribution than the unit cost function, which means that unit action cost can benefit more from tie-breaking. On the other hand, cost functions that contain 0 cost may induce a distance distribution that is even more concentrated than that of the unit cost function, suggesting actions with 0 cost may bring some advantages through tie-breaking. Nevertheless, best-case tie-breaking cannot be guaranteed when 0 cost actions exist, and they may cause "g-value" plateaus which increase the number of nodes expanded (Benton et al. 2010).

Hazardous Logistics

While unit action costs *often* produce a more concentrated distance distribution than non-unit action cost, this is not always the case. We construct a transition graph structure

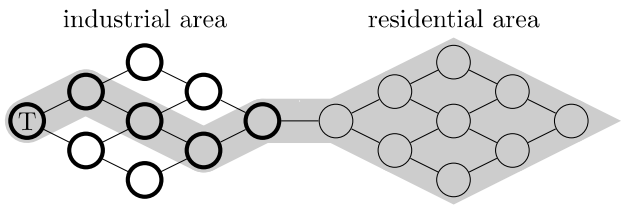

Figure 8: An illustration for hazardous logistics. Thicker circles represent industrial locations and thinner circles represent residential locations. The gray area indicates locations a residential mode truck can visit. A truck starts at the leftmost location (the circle containing a "T").

where the non-unit cost function induces the more concentrated distribution.

Consider a logistics-like problem where the locations are grouped into residential and industrial regions. In industrial regions, hazardous products may be transported. Each truck can be in one of three modes: unassigned, residential and industrial. An unassigned truck cannot move, but can be assigned to residential or industrial mode at most once (to avoid the risk of mixing hazardous and safe products). An industrial mode truck is restricted to move within the industrial region. A residential mode truck can only visit a corridor subset of industrial locations. For simplicity, we consider only moving a truck, without transporting any product. See Figure 8 for an illustration. In the initial state, the truck is at the leftmost location of the map in the unassigned mode. The first step for using this truck is to make a choice between industrial and residential mode. If the truck is changed to industrial mode, then it can move only in the industrial area. If the truck is changed to residential mode, then it first moves across the industrial region and then in the residential area. In different modes, the truck generates

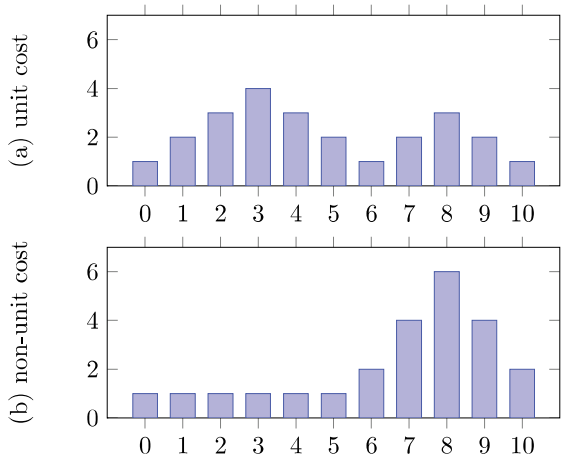

Figure 9: For the example of Figure 8, distance distribution induced by unit cost function (a) is less concentrated than that of non-unit cost function (b).

different sets of states. With unit cost, the reachable states correspond to two peaks in a bimodal distribution since the residential area is further away (Figure 9 (a)). However, if we assign a cost of 6 for changing the truck into industrial mode, then we get a more concentrated distribution as the two peaks in unit cost become a single peak in a unimodal distribution (Figure 9 (b)). The sum-of-squares is 62 for the unit cost function, and 82 for non-unit cost.

Conclusion

We have presented substantial empirical evidence to show that diverse cost functions *can* be beneficial for search. In particular, increasing the cost of an action can have either a positive or a negative effect on a given problem instance: it can lead to additional search by delaying the application of an expensive but necessary action, but it can also accelerate the search by (temporarily) blocking irrelevant actions and making a path to a goal more attractive to search.

While analyzing single instances can be useful in practice, we have shown that the preference of one cost function over another largely disappears when averaging over problem instances within the same state space. Our No Free Lunch theorem makes this claim precise. Furthermore, we have analyzed the effect of tie-breaking, and shown that its effect on search efficiency is controlled by the concentration of the distribution of path costs. Unit costs often have better concentration than non-unit costs but we have introduced a new planning domain, Hazardous Logistics, in which the opposite is true.

This work raises an important question for satisficing or cost-bounded search. Current systems use components based on unit costs to speed up search. Our results show that there might be better cost functions for this purpose than unit costs.

We can imagine two possible ways in which the results of this paper might benefit cost-optimal planning. Both are speculative, we mention them as possible directions for future research. The first is that depth-based tie-breaking strategies (Asai and Fukunaga 2016; 2017) use distances based on unit costs to guide the search in the final plateau of domains with abundant zero-cost actions. Our work suggests that distances based on non-unit costs could also be used in such tie-breaking strategies and have the potential to achieve better performance. The second possible application is to cost-optimal algorithms that make use of an upper bound on optimal solution cost, such as breadth-first heuristic search (Zhou and Hansen 2004). It is possible they might be able to quickly find a good bound using an alternative cost function.

Acknowledgments

The authors gratefully acknowledge the funding from the Natural Sciences and Engineering Research Council of Canada.

References

Aghighi, M., and Bäckström, C. 2015. Cost-optimal and net-benefit planning - A parameterised complexity view. In

Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI-2015, 1487–1493.

Aghighi, M., and Bäckström, C. 2016. A multi-parameter complexity analysis of cost-optimal and net-benefit planning. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling, ICAPS-2016*, 2–10.

Asai, M., and Fukunaga, A. S. 2016. Tiebreaking strategies for A* search: How to explore the final frontier. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence, AAAI-2016*, 673–679.

Asai, M., and Fukunaga, A. 2017. Tie-breaking strategies for cost-optimal best first search. *Journal of Artificial Intelligence Research (JAIR)* 58:67–121.

Benton, J.; Talamadupula, K.; Eyerich, P.; Mattmüller, R.; and Kambhampati, S. 2010. G-value plateaus: A challenge for planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS-2010*, 259–262.

Cushing, W.; Benton, J.; and Kambhampati, S. 2010. Cost based search considered harmful. In *Proceedings of the 3rd Annual Symposium on Combinatorial Search, SoCS-2010*, 140–141.

Cushing, W.; Benton, J.; and Kambhampati, S. 2011. Cost based satisficing search considered harmful. *CoRR* abs/1103.3687.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1(1):269–271.

Edelkamp, S., and Schrödl, S. 2012. *Heuristic Search - Theory and Applications*. Academic Press.

Ikeda, T.; Hsu, M.-Y.; Imai, H.; Nishimura, S.; Shimoura, H.; Hashimoto, T.; Tenmoku, K.; and Mitoh, K. 1994. A fast algorithm for finding better routes by AI search techniques. In *Proceedings of Vehicle Navigation and Information Systems Conference*, 291–296.

Katz, M., and Domshlak, C. 2008. New islands of tractability of cost-optimal planning. *Journal of Artificial Intelligence Research (JAIR)* 32:203–288.

Martelli, A. 1977. On the complexity of admissible search algorithms. *Artificial Intelligence* 8(1):1–13.

Nakhost, H. 2013. *Random Walk Planning: Theory, Practice, and Application*. Ph.D. Dissertation, University of Alberta.

Pearl, J. 1984. *Heuristics – Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research (JAIR)* 39:127–177.

Richter, S.; Westphal, M.; and Helmert, M. 2011. Lama 2008 and 2011. In *International Planning Competition*, 117–124.

Thayer, J. T., and Ruml, W. 2009. Using distance estimates in heuristic search. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS-2009*, 382–385.

Thayer, J. T., and Ruml, W. 2011. Bounded suboptimal search: A direct approach using inadmissible estimates. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI-2011*, 674–679.

Thayer, J. T.; Benton, J.; and Helmert, M. 2012. Better parameter-free anytime search by minimizing time between solutions. In *Proceedings of the 5th Annual Symposium on Combinatorial Search, SoCS-2012*, 120–128.

Wilt, C. M., and Ruml, W. 2011. Cost-based heuristic search is sensitive to the ratio of operator costs. In *Proceedings of the 4th Annual Symposium on Combinatorial Search, SoCS-2011*, 172–179.

Wilt, C. M., and Ruml, W. 2014. Speedy versus greedy search. In *Proceedings of the Seventh Annual Symposium on Combinatorial Search, SoCS-2014*, 184–192.

Wilt, C. M. 2014. *Steps Towards a Science of Heuristic Search*. Ph.D. Dissertation, University of New Hampshire.

Wolpert, D., and Macready, W. 1995. No free lunch theorems for search. Technical report, Santa Fe Institute.

Wolpert, D., and Macready, W. G. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1(1):67–82.

Zhou, R., and Hansen, E. A. 2004. Breadth-first heuristic search. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling, ICAPS-2004*, 92–100.