Planning with Multiple Action-Cost Estimates

Eyal Weiss, Gal A. Kaminka

The MAVERICK Group, Bar-Ilan University, Israel
eyal.weiss@biu.ac.il, galk@cs.biu.ac.il

Abstract

AI Planning require computing the costs of ground actions. While often assumed to be negligible, the run-time of this computation can become a major component in the overall planning run-time. To address this, we introduce *planning with multiple action cost estimates*, a generalization of classical planning, where action cost can be incrementally determined using multiple estimation procedures, which trade computational effort for increasingly tightening bounds on the exact cost. We then present ACE, a generalized A^*, to solve such problems. We provide theoretical guarantees, and extensive experiments that show considerable run-time savings compared to alternatives.

Introduction

AI planning applications require computing the costs of ground actions to optimize plans. The costs are computed by the planner for concrete ground actions that it considers adding to the plan, as part of the planning process. Typically, they are computed from a domain action model provided as input to the planner (e.g., using PDDL (McDermott et al. 1998; Fox and Long 2003)). The run-time of action cost computation is generally considered to be negligible.

However, the computation of action costs can incur significant computational expense, to the point it becomes a major component in the overall planning run-time. This happens when the computation is inherently expensive, e.g., as in the case of robot task and motion planning, where the planning process considers geometric and kinematic constraints on movement in continuous environments, and repeatedly computes distances and potential collisions (LaValle 2006; Garrett et al. 2021; Dellin and Srinivasa 2016; Narayanan and Likhachev 2017; Mandalika et al. 2019). Cost computation run-time can also increase by the use of external blackbox action models (including action costs) that are called by the planner as needed (Dornhege et al. 2009; Gregory et al. 2012; Francès et al. 2017; Allen et al. 2021).

Transportation logistics planning is an example domain where the two sources of expensive computation of action costs combine in real-world applications. The domain deals with planning efficient routes for trucks and payloads. A simple form of it is captured in the *Transport Sequential* IPC domain (Coles et al. 2012), where costs are assumed to be given in the PDDL problem description, and their computation is immediate. But in actual commercial applications (e.g., (TruckNet 2021)), *the route cost computation is expensive in itself*, as the system takes into account also truck maintenance and insurance costs, driver salaries, fuel prices in different locations, legal speed limits, time-of-day, and geometrical constraints such as elevation and road curvature. The travel time between cities can also be computed by queries to an online service such as Google Maps, an *external action model*. The online query time is measured in milliseconds per query; far from negligible.

To reduce the computational expense in computing action costs, previous work explored mainly two approaches. The first focuses on minimizing the number of independent expensive cost computations that must be carried out (Dellin and Srinivasa 2016; Narayanan and Likhachev 2017; Mandalika et al. 2019). The second, exemplified by the D^* (Stentz 1994)) and *Lifelong Planning* A^* (Koenig, Likhachev, and Furcy 2004) algorithm families, minimizes cost computations when previously-computed costs change.

We propose an alternative approach building on the key insight that the computation of action costs can be spread across *multiple* procedure calls. A first rough estimate can be generated quickly, while future calls to cost computation procedures can increasingly improve the estimation, potentially taking longer to compute. This approach follows the line of the recently suggested concept of dynamic estimation during planning (Weiss 2022; Weiss and Kaminka 2022).

In the case of the transportation logistics example, the travel time between two cities can be estimated quickly—but inaccurately—from their fixed geographical distance or mean travel times computed once, offline (e.g., the way a *transport sequential* problem file includes fixed costs). Online queries and expensive computations of travel time based on elevation and curvature can serve as more accurate—and more costly—estimates. More parameters can be taken into account in increasingly more expensive estimator procedures, resulting in more accurate estimates.

We therefore define a novel planning problem, *planning with multiple action cost estimates* (P-MACE). This is a generalization of classical planning problems, where every ground action can have multiple cost estimators. Each estimator provides a lower and upper bound on the actual cost, and the estimators are ordered by increasing computational expense. A P-MACE planner's task is to efficiently find a plan *meeting a target bound on the optimal cost*. This definition generalizes previous work in motion planning (Dellin and Srinivasa 2016); see details in a separate section.

We present ACE (A^* with Cost Estimation), a generalization of A^* (Hart, Nilsson, and Raphael 1968) that allows a planner to efficiently select estimators so to generate plans meeting the target bound. ACE is sound (finds a correct plan) and complete (will always find a plan if one exists). Its optimality is not dependent only on the heuristic used, but also on the available estimators. In case every action cost can ultimately be estimated exactly, ACE is optimal. In other cases, it may fail to find plans meeting the target bound. To augment ACE, we present a post-search procedure that improves the accuracy of non-optimal solutions that do not meet the target bound. We implemented ACE in the Fast Downward planner (Helmert 2006). Extensive experiments with 6600 planning problems generated from IPC benchmarks shows dramatic reduction in run-time compared to available alternatives, while maintaining estimated costs well within target bounds. The savings are also clear in an experiment with real-world data for the Ontario province (Canada) using Google Maps query times.

Preliminaries

We use a simple notation for deterministic planning based on finite-state automaton (Ghallab, Nau, and Traverso 2016). A *planning domain* is a 4-tuple $\Sigma = (S, A, \gamma, c)$, where S is a finite set of system states, A is a finite set of (ground) actions, $\gamma : S \times A \to S$ is a partial function called the state-transition function, and $c : S \times A \to \mathbb{R}^+$ is a partial function called the cost function, that has the same domain as γ. If $\gamma(s,a) = s'$ is defined, then a is said to be applicable in s, and its cost $c(s,a) < \infty$ is called the transition cost. For simplicity we take $c(s,a)$ to be $c(a)$, i.e., we consider the costs of ground actions rather than transition costs. This is common practice and serves here for easier presentation. Nevertheless, the techniques presented below fully support dealing with transition costs.

Definition 1. *A* **classical planning problem** *is a triple $\mathcal{P} = (\Sigma, s_0, S_g)$, where Σ is a planning domain, $s_0 \in S$ is an initial state, and $S_g \subset S$ is a set of goal states. A* **solution** *for $\mathcal{P}$ is a plan $\pi = \langle a_1, ..., a_n \rangle$, a finite sequence of actions, s.t. $a_1 \in \pi$ is applicable in s_0; for all $i \leq n$, a_i is applicable in s_{i-1} and $s_i = \gamma(s_{i-1}, a_i)$; and $s_n \in S_g$. The plan length is $|\pi| = n$, and its cost is $c(\pi) = \sum_{i=1}^{n} c(a_i)$. A* **cost-optimal solution** *is a plan π^* that satisfies*

$$c^* := c(\pi^*) = \min\{c(\pi') \mid \pi' \textit{ is a solution for } \mathcal{P}\}.$$

The directed graph $\mathcal{G}_\Sigma = (\mathcal{V}, \mathcal{E}, w)$ induced by the planning domain Σ is a *weighted digraph*, where: $\mathcal{V}$ is a set of vertices, each corresponding to a state $s \in S$, and $\mathcal{E}$ is a set of edges, s.t. $e = (s, \gamma(s,a)) \in \mathcal{E}$ iff an action $a \in A$ is applicable in s. For each edge $e \in \mathcal{E}$ the weight (cost) is given by $w(e) = c(a)$. For simplicity, we use $c(e)$ for edge weights.

We note that in the general case, it may be that two distinct actions a_1, a_2 applied in a state s both result in the same state, $\gamma(s, a_1) = \gamma(s, a_2)$, i.e., multiple edges connect the same source and target vertices, which would make $\mathcal{G}_\Sigma$ a multi-graph. However, for simplicity of the presentation, we consider the planning domains in the paper to have the property that every two actions applied in the same state result in two distinct states. Therefore, there is a one-to-one correspondence between a planning domain Σ and the graph $\mathcal{G}_\Sigma$.

Planning with Action Cost Estimators

This section introduces mathematical definitions and notations for discussing action cost estimators. We then pose a novel planning problem that considers multiple cost estimators, generalizing over problems with exact costs. We conclude with a theoretical analysis that lays the basis for developing algorithms that solve the newly suggested problem.

Multiple Action-Cost Estimates

Mathematical Setting. We begin by associating a set of cost estimators with every ground action in a planning domain Σ (each edge e in $\mathcal{G}_\Sigma$):

Definition 2. *The* cost estimators function Θ *for a planning domain Σ maps each edge $e \in \mathcal{E}$ to a finite and non-empty ordered set of estimators,*

$$\Theta(e) = (\theta_e^1, \ldots, \theta_e^{k(e)}), k(e) \in \mathbb{N},$$
$$\textit{s.t., } \forall e, i, \theta_e^i : \emptyset \to \mathbb{R}^+ \times \mathbb{R}^+,$$

where each estimator θ_e^i defines an accuracy interval, and $\Theta(e)$ is ordered by increasing estimation times (which are guaranteed to be finite).

Throughout this paper we make the following assumptions about cost estimators functions:

1. Each estimator provides finite bounds for the true cost, i.e., $\theta_e^i = (c_{min}^i(e), c_{max}^i(e))$ with $0 \leq c_{min}^i(e) \leq c(e) \leq c_{max}^i(e) < \infty$.
2. Positive costs have positive bounds, namely $c(e) > 0$ implies $c_{min}^i(e) > 0$.
3. Estimation bounds tighten with increasing order of estimators, so that the intervals induced by θ_e^j, θ_e^i satisfy $[c_{min}^j(e), c_{max}^j(e)] \subseteq [c_{min}^i(e), c_{max}^i(e)], \forall e, i < j$.

We note that zero costs are allowed, and that Assumptions 1 and 2 together imply that zero costs are identified by estimators, i.e., $c(e) = 0$ iff $c_{min}^i(e) = 0$. We also note that Assumption 1 can be relaxed to allow infinite values, and Assumption 2 can be removed entirely, yet they both simplify the paper's exposition.

The Planning Problem. The definition of planning changes when Θ is considered. The notion of a solution plan remains (sequence of actions from s_0 to a goal state). However, the optimal plan cost, that depends on exact costs, may not be computable. This is because Def. 2 and Assumptions 1–3 allow the exact cost of any edge to remain unknown, even after utilizing all possible estimates for that cost. Hence, we define optimality w.r.t. a bound $\mathcal{B}$ on the optimal (possibly unknown) cost, so estimation uncertainty is taken into account. This is formalized as follows.

Definition 3. *A* P-MACE **problem** *is a tuple* $\mathcal{P} = (\Sigma, \Theta, s_0, S_g)$*, where* Σ *is a planning domain with cost function* $c \in \Sigma$ *unknown,* Θ *is a cost estimators function for* Σ*,* $s_0 \in S$ *is an initial state, and* $S_g \subset S$ *is a set of goal states. A* $\mathcal{B}$**-optimal solution** *to* $\mathcal{P}$ *is a plan* $\pi^{\mathcal{B}}$ *that satisfies*

$$c(\pi^{\mathcal{B}}) \leq c^* \times \mathcal{B},$$

with c^* *being the optimal cost and* $\mathcal{B} \geq 1$.

P-MACE generalizes classical planning (Thm. 1) in two aspects: first, it allows the cost of an action to be unknown, yet quantified by a bounding interval; and second, it permits multiple time-consuming cost estimators per action. The implications of latter generalization is that planners can exploit this flexibility when searching for a solution by spreading estimation time across cheaper alternatives, when lower accuracy is deemed sufficient, in order to save run-time.

Theorem 1 (Generality)**.** P-MACE *problems are a generalization of classical planning problems.*

Proof. Any classical planning problem can be formulated as a P-MACE problem, by considering the special case where each edge has one estimator (i.e., $k(e) = 1$ for every e), that returns the true cost (namely, $c^1_{min}(e) = c(e) = c^1_{max}(e)$), with no relaxation for plan optimality (which means that $\mathcal{B} = 1$). □

Plan Cost Uncertainty Quantification

Naively, $\mathcal{B}$-optimal planning can be carried out by an *estimation-indifferent* process, that ignores computational expense and evaluates all estimators for any action a it considers for a solution. However, unless estimation time is negligible, the planner should instead carefully select the estimators to be applied.

Let Θ_Σ be the set of all estimators, for all edges in $\mathcal{G}_\Sigma$. A subset Φ of Θ_Σ is *valid* if it contains at least one estimator per edge. Given a plan $\pi = \langle a_1, ..., a_n \rangle$ and a valid $\Phi \subseteq \Theta_\Sigma$, the *plan lower bound* and *plan upper bound* w.r.t. Φ are defined (respectively) as

$$c^\Phi_{min}(\pi) := \sum_{i=1}^{n} c^i_{min,\Phi},\ c^\Phi_{max}(\pi) := \sum_{i=1}^{n} c^i_{max,\Phi}, \qquad (1)$$

where $c^i_{min,\Phi}$ and $c^i_{max,\Phi}$ denote the *tightest* lower and upper bound estimates of a_i obtained from Φ. The cost uncertainty ratio of π w.r.t. Φ is then

$$\eta(\pi) := \frac{\sum_{i=1}^{n} c^i_{max,\Phi}}{\sum_{i=1}^{n} c^i_{min,\Phi}} = \frac{c^\Phi_{max}(\pi)}{c^\Phi_{min}(\pi)} \qquad (2)$$

We define $\eta(\pi)$ to be 1 in case $\sum_{i=1}^{n} c^i_{min} = 0$, as this implies $c(\pi) = 0$, meaning that there is no uncertainty.

The choice of Φ directly impacts the cost uncertainty of a plan π, since

$$c^\Phi_{min}(\pi) \leq c(\pi) \leq c^\Phi_{max}(\pi) = c^\Phi_{min}(\pi) \times \eta(\pi) \qquad (3)$$

The *optimal* plan cost lower bound $c^{*\Phi}_{min}$, and the *optimal* plan cost upper bound $c^{*\Phi}_{max}$ are the minimal plan cost lower and upper bounds over all plans for $\mathcal{P}$. An *optimal optimistic plan* π_{opt} and an *optimal pessimistic plan* π_{pes} are plans that satisfy $c^\Phi_{min}(\pi_{opt}) = c^{*\Phi}_{min}$, $c^\Phi_{max}(\pi_{pes}) = c^{*\Phi}_{max}$.

Thm. 2 shows that the cost of an *optimal* optimistic plan is bounded by its cost uncertainty ratio and the optimal (unknown) cost c^*. Corollary 1 then shows this leads to being able to efficiently find $\mathcal{B}$-optimal plans, by searching for effective Φ.

Theorem 2 (Bound)**.** *Given a* P-MACE *problem* $\mathcal{P}$*, an optimal optimistic plan* π_{opt} *w.r.t. any valid* $\Phi \subseteq \Theta_\Sigma$*, satisfies the following relation for its cost:*

$$c(\pi_{opt}) \leq c^* \times \eta(\pi_{opt}). \qquad (4)$$

Proof. Denote by (c^i_{min}, c^i_{max}) the lower and upper bounds, for the cost of the ith action in π_{opt}. By definition of π_{opt} as an optimal optimistic plan w.r.t. Φ, it follows that

$$\sum_{i=1}^{n} c^i_{min} \leq c^*. \qquad (5)$$

From the right inequality of (3), we get

$$c(\pi_{opt}) \leq \sum_{i=1}^{n} c^i_{max} = \sum_{i=1}^{n} c^i_{min} \times \eta(\pi_{opt}). \qquad (6)$$

Using Inequality (5) to obtain an upper bound of the right hand side of Inequality (6) yields Inequality (4). □

Corollary 1. *It follows from Thm. 2 that an optimal optimistic plan* π_{opt} *w.r.t. any valid* $\Phi \subseteq \Theta_\Sigma$ *with* $\eta(\pi_{opt}) \leq \mathcal{B}$ *is* $\mathcal{B}$*-optimal. This implies that we can reduce planning time by efficiently finding* Φ *that achieves this.*

ACE: A^* with Cost Estimation

We present ACE, a generalization of A^*, that solves P-MACE problems while reducing the number of unnecessary cost estimations. Based on Corollary 1, ACE uses the target sub-optimality factor $\mathcal{B}$ to restrict the number of estimates used during the search for plans, comparing $\mathcal{B}$ to $\eta(p)$ for every path p being considered.

ACE (Alg. 1) is given a P-MACE problem $\mathcal{P}$, a target $\mathcal{B}$, a heuristic h and a procedure GetEstimator, which incrementally selects estimators $\theta^i_e \in \Theta(e)$ for an edge e (see details below). ACE then tries to find a $\mathcal{B}$-optimal plan, and outputs the plan π it finds and reports its achieved cost uncertainty ratio $\eta(\pi)$. If no plan is found, $(\emptyset, \infty)$ is returned.

We explain how ACE works by comparing it to A^*. ACE uses accumulated bounds g_{min} and g_{max}, instead of the accumulated cost g. Similarly, f-values are computed by $f(s) = g_{min}(s) + h(s)$, (i.e., g_{min} replaces g). There are two important deviations from A^*: in computing the heuristic values, and in estimating the cost of edges.

First, ACE deviates from A^* in the costs used for the computation of heuristic h values. In A^*, these are the edge (action) cost values. ACE only has access to cost bounds instead of exact costs, so the h-values are computed based on lower bounds. Specifically, the *loosest* (lowest) lower bound for each edge is utilized, to preserve heuristic consistency (a straightforward minor result; see Lemma 2 in the appendix).

Algorithm 1: ACE

```
Input: Problem $\mathcal{P} = (\Sigma, \Theta, s_0, S_g)$, target $\mathcal{B}$
Parameter: Heuristic $h$, procedure GetEstimator
Output: Plan $\pi$, bound $\eta$
1:  $g_{min}(s_0) \leftarrow 0$; $g_{max}(s_0) \leftarrow 0$
2:  OPEN $\leftarrow \emptyset$; CLOSED $\leftarrow \emptyset$
3:  Insert $s_0$ into OPEN with $f(s_0) = h(s_0)$
4:  while OPEN $\neq \emptyset$ do
5:      $n \leftarrow$ best node from OPEN
6:      if $Goal(n)$ then
7:          return $trace(n), g_{max}(n)/g_{min}(n)$
8:      Insert $n$ into CLOSED
9:      for each successor $s$ of $n$ do
10:         if $s$ not in OPEN $\cup$ CLOSED then
11:             $g_{min}(s) \leftarrow \infty$
12:         $\eta \leftarrow \infty$; $\underline{g} \leftarrow g_{min}(n)$
13:         $\theta \leftarrow$ GetEstimator$(e = (n, s))$
14:         while $\eta > \mathcal{B}$ and $\underline{g} < g_{min}(s)$ and $\theta \neq \emptyset$ do
15:             $\underline{c}, \bar{c} \leftarrow$ apply$(\theta)$
16:             $\underline{g} \leftarrow g_{min}(n) + \underline{c}$; $\bar{g} \leftarrow g_{max}(n) + \bar{c}$
17:             $\eta \leftarrow \bar{g}/\underline{g}$
18:             $\theta \leftarrow$ GetEstimator$(e)$
19:         if $\underline{g} < g_{min}(s)$ then
20:             $g_{min}(s) \leftarrow \underline{g}$; $g_{max}(s) \leftarrow \bar{g}$
21:             if $s$ in OPEN $\cup$ CLOSED then
22:                 Remove $s$ from OPEN and CLOSED
23:             Insert $s$ into OPEN with $f(s) = g_{min}(s) + h(s)$
24: return $\emptyset, \infty$
```

The second deviation of ACE from A^* is that instead of simply using $c(e)$ to calculate $g(s)$ for a node s, an estimation loop is introduced (lines 13–18) to acquire estimated cost bounds. For the edge e connecting nodes n and s ($e = (n, s)$, line 13), the loop iterates over possible estimators until the bound $\mathcal{B}$ is met by η of the path ending in node s, or an alternative path with lower $g_{min}(s)$ was already found, or no estimators are left. In each iteration, line 15 computes $\underline{c}$ (the *tightest* lower bound for e over all estimators applied so far), and $\bar{c}$ (similarly, the *tightest* upper bound). When the loop is done, if a path with lower $g_{min}(s)$ is found, then $g_{min}(s)$, $g_{max}(s)$ are updated (lines 19–20).

Every call to GetEstimator(e) returns the next estimator from the set $\Theta(e)$ which has yet to be applied, or $\emptyset$ when none are left. ACE's theoretical guarantees (below) assume nothing about the order of the estimators selected by GetEstimator. However, to save run-time, $\Theta(e)$ is ordered by increasing run-time. Thus, each time GetEstimator is invoked on the edge e, it returns the next estimator θ_e^i with the shortest run-time. This is the approach we took in the experiments.

Remark 1. *Note that an estimation-indifferent planning algorithm is easily derived from Alg. 1, by modifying it to only consider the tightest bounds.*

Analysis of ACE: Guarantees

We start by proving the completeness of ACE, which follows almost immediately from the structure of A^*, inherited by ACE.

Theorem 3 (Completeness)**.** ACE *always terminates and returns a plan when one exists.*

Proof sketch. First, the fact that ACE always terminates follows from the same reasoning that A^* always terminates, together with the fact that there is a finite number of estimators per edge, each with finite run-time. Second, in order to prove that ACE terminates with a plan, when one exists, we can again rely on the same reasoning as in the proof of A^*, where we only have to show that each new node encountered during the search is inserted into OPEN. Every new node (generated in line 9) triggers $g_{min}(s) \leftarrow \infty$ (line 11). In line 12 $\underline{g} \leftarrow g_{min}(n)$, namely $\underline{g}$ is initialized to some finite number. Then, in lines 14–18 $\underline{g}$ may be updated again, but it is guaranteed to remain a finite number, since every lower bound returned by an estimator (indicated as $\underline{c}$) is also finite. Thus, the condition in line 19 is met, and so s is necessarily inserted to OPEN (line 23). □

Next, we show that ACE is sound, i.e., if it returns a plan then it must be correct. To do this, we rely on Lemma 1, stated as follows (see appendix for full proof).

Lemma 1 (Φ Set Optimality)**.** *Provided with a consistent heuristic h,* ACE *necessarily returns an optimal optimistic plan w.r.t. some $\Phi \subseteq \Theta_\Sigma$, if a plan exists.*

Proof sketch. The proof relies on three arguments:

1. That heuristic consistency is preserved under lower bounds of edge costs (Lemma 2 in the appendix).
2. Recognizing $\Phi \subseteq \Theta_\Sigma$ as the set that includes exclusively all estimators invoked by ACE during the search.
3. Following the proof arguments of A^*'s optimality (that relies on a consistent heuristic) to establish that whenever ACE selects a node for expansion, an optimal path (w.r.t. the estimates utilized up to that point) to that node has been found.

Combining the arguments above proves that ACE, together with a consistent heuristic h, returns an optimal plan w.r.t. the lower bound estimates provided by estimators in Φ. □

In P-MACE, soundness needs to be defined in terms of the bound requested of the solution. We define this property as follows. An algorithm is said to be $\mathcal{B}$**-sound** if every time it returns a plan reported as $\mathcal{B}$-optimal, the plan returned is indeed $\mathcal{B}$-optimal. Thm. 4 shows this is true of ACE.

Theorem 4 ($\mathcal{B}$-Soundness)**.** *Provided with a consistent heuristic h,* ACE *is $\mathcal{B}$-sound.*

Proof of Thm. 4. By induction on the order of nodes entering OPEN, starting from the base case of the start node, we show that each node n in OPEN satisfies $\eta(n) := g_{max}(n)/g_{min}(n)$ (where the case of $g_{min}(n) = 0$ is considered $\eta(n) = 1$, as there is no uncertainty). Thus, if a plan π is found, terminating at s_g, it necessarily means that

$\eta(s_g) = \eta(\pi)$ is returned (i.e., the η returned is correct). In addition, according to Lemma 1, using a consistent h ensures that π is guaranteed to be an optimal optimistic plan w.r.t. some valid $\Phi \subseteq \Theta_\Sigma$. Hence, if $\eta(\pi) \leq \mathcal{B}$ is satisfied, then relying on Thm. 2, π is a $\mathcal{B}$-optimal plan, whereas in case it is not satisfied, then π is not guaranteed by ACE to be $\mathcal{B}$-optimal. □

We now transition to discuss the optimality of ACE. To do this, we first clarify the following.

$\mathcal{B}$-Optimality: An algorithm is said to be *$\mathcal{B}$-optimal* if it is guaranteed to find a plan that can shown to be a $\mathcal{B}$-optimal plan (Def. 3) using estimators in Θ.

In the general case ACE is not $\mathcal{B}$-optimal. Not every plan returned will meet the target bound. It might even be that in generating the resulting plan, not all estimation options are exhausted, thus ACE could return a plan that is not $\mathcal{B}$-optimal, though one exists.

However, under some conditions, it is $\mathcal{B}$-optimal (Thm. 5). Intuitively, if every edge can be estimated to the desired degree of certainty, then starting from a bound on the path cost uncertainty which is under the threshold, and adding only edges with "good enough" cost estimates, it is possible to retain all paths explored with accumulated η lower or equal to $\mathcal{B}$. Then, the first solution found (if it exists) necessarily meets the requirement.

Theorem 5 (Special $\mathcal{B}$-Optimality). *Given a* P-MACE *problem $\mathcal{P}$ with sub-optimality target $\mathcal{B}$, if every edge $e \in \mathcal{E}$ satisfies $c^i_{max}(e)/c^i_{min}(e) \leq \mathcal{B}$ (or $c^i_{min}(e) = c(e) = 0$) for some i (that can be different for each edge), and a consistent heuristic h is used, then* ACE *is $\mathcal{B}$-optimal.*

Proof of Thm. 5. First, that fact that ACE is complete follows from Thm. 3. Second, similar to the proof of Thm. 4, and using the fact that for every edge e the bound $c^i_{max}(e)/c^i_{min}(e) \leq \mathcal{B}$ (or the equality $c^i_{min}(e) = c(e) = 0$) can be achieved (for some i), it can be shown by induction that each node n in OPEN satisfies $\eta(n) = g_{max}(n)/g_{min}(n) \leq \mathcal{B}$ (where again, $g_{min}(n) = 0$ is considered as $\eta(n) = 1$). Therefore, if a $\mathcal{B}$-optimal plan exists, then ACE will necessarily return such one. □

It follows from Thm. 5 that ACE **is $\mathcal{B}$-optimal in the case of classical planning**, where an exact-cost estimate is available for every edge. See below.

Corollary 2. *If the cost of every edge can ultimately be estimated exactly, i.e., $\forall e \in \mathcal{E}$ it holds that $\exists i : c^i_{max}(e) = c^i_{min}(e)$, then necessarily $c^i_{max}(e)/c^i_{min}(e) = 1 \leq \mathcal{B}$, i.e.,* ACE *is $\mathcal{B}$-optimal (Thm. 5).*

Post-Search Improvements

We provide a simple post-search procedure (Proc. 1) that aims to reduce the cost uncertainty $\eta(\pi)$ when ACE terminates with $\eta(\pi) > \mathcal{B}$. We observe that even then, there may still be unused estimators for edges (ground actions) along the path (plan) found. It therefore may still be possible to improve $\eta(\pi)$ by applying additional estimators for edges that are already in π.

Procedure 1: End of Search Estimations (ESE)

Input: ACE's inputs and variables before termination
Parameter: Procedure GetEstimator
Output: Bound η

```
1:  for each edge $e$ that corresponds to an action from $\pi$ do
2:      $\theta \leftarrow$ GetEstimator(e)
3:      while $\eta > \mathcal{B}$ and $\theta \neq \emptyset$ do
4:          $\underline{c}, \bar{c} \leftarrow$ apply($\theta$)
5:          update $c_{max}(\pi)$ using $\bar{c}$
6:          $\eta \leftarrow c_{max}(\pi)/c_{min}(\pi)$
7:          $\theta \leftarrow$ GetEstimator(e)
8:      if $\eta \leq \mathcal{B}$ then
9:          return $\eta$
10: return $\eta$
```

The basic idea is to loop over all the edges along the path corresponding to the plan found, and for each one try to reduce its uncertainty by applying unused estimators. We emphasize that the tightest lower bound we may use to calculate $\eta(\pi)$, so that π can still be guaranteed to be optimal optimistic w.r.t. some Φ (which is needed in order to relate $\eta(\pi)$ to $\mathcal{B}$, due to Thm. 2), cannot be greater than the one already found. The reason is that a different plan $\tilde{\pi}$ might exist with $c_{min}(\tilde{\pi})$ equal to that f-value. For this reason only upper bounds are updated in the procedure.

Empirical Evaluation

The theoretical properties of ACE give little insight as to its run-time behavior and success rate. ESE does not carry theoretical guarantees. We therefore focus on extensive experiments to evaluate ACE and ESE in a variety of benchmark planning problems where we control the need for estimation and target sub-optimality bound.

We developed *PlanDEM* (Planning with Dynamically Estimated Action Models), a planner that provides a concrete implementation of ACE and ESE, in C++. PlanDEM (Weiss and Kaminka 2023) modifies and extends Fast Downward (FD) (Helmert 2006) (v20.06), and inherits its use of the h_{max} heuristic (Bonet and Geffner 2001), used in the experiments (where h_{max} was chosen due to its consistency). The underlying FD search algorithm and data structures were modified appropriately. All experiments were run on an Intel i7-6600U CPU (2.6GHz), with 16GB of RAM, in Linux.

We modified standard planning problems from past planning competitions, to include synthetic estimators at several levels of uncertainty. Specifically, we selected 20 domains & problems with action costs (Fox and Long 2003). For each, we generated a random variant with a target $\mathcal{B}$, and varying the number of actions with estimated (rather than precise) costs with a probability p_1. When we set $p_1 = 1$, the costs of all ground actions (edges in $\mathcal{G}_\Sigma$) are estimated by the planner. When $p_1 = 0$, all costs are given precisely. When $0 < p_1 < 1$, only some ground actions require estimation of the cost. Thus a pair $\mathcal{B}, p_1$ creates a new instance of a benchmark problem. The costs, their bounds and names of domains & problems used are in the technical appendix.

Each estimated-cost edge e with precise cost $c(e)$ is provided with a set of three estimators of $c(e)$, $\Theta(e) = \{\theta_e^1, \theta_e^2, \theta_e^3\}$. These have $c_{max}^i(e)/c_{min}^i(e)$ uncertainty ratios of 4, 2, and 1 (the correct value), resp.

Since the problems we address have a target bound $\mathcal{B}$, but no other solution ranking criteria, we choose to prefer time over smaller uncertainty as a secondary ranking criterion. Given two $\mathcal{B}$-optimal solutions π_1, π_2 with corresponding $\eta^1 < \eta^2$, but run-times $t^1 > t^2$, we consider π_2 to be better.

Evaluation of ACE

We take p_1 to be one of $\{0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1\}$ and $\mathcal{B}$ values in the range $[1, 4]$ in jumps of 0.25, resulting in a total of 1820 runs. Note that $\mathcal{B} = 1$ requires precise cost, while $\mathcal{B} = 4$ is equivalent in our setting to having no requirement at all on precision (as the highest uncertainty ratio given by an estimator is 4). As one of the estimators has uncertainty ratio of 1, the cost of each edge can be eventually estimated perfectly, and thus Thm. 5 holds (i.e., the achievable η is 1). Therefore, ACE will always terminate with a $\mathcal{B}$-optimal solution.

Comparison to Baseline. As the planning problem defined in this paper is new, there is no immediate baseline for comparison other than the naive *estimation-indifferent* planning utilizing all available estimators.

We empirically contrast PlanDEM with estimation-indifferent planning. We begin by examining ACE's run-time behavior from the perspective of estimators utilization. In particular, as every edge considered has to be estimated at least once, we focus on the number of expensive estimators, i.e., (those with tighter ratios 2, 1).

Fig. 1 plots the ratio between the actual number of expensive estimations that ACE used and the maximum number of potential expensive estimations on the vertical axis, against the target $\mathcal{B}$ bound. Thus a lower ratio indicates an improved result, i.e., less expensive estimations used. The estimation-indifferent planning approach always uses all estimators, and therefore its ratio is always 1, indicated by the the straight solid horizontal black line at the top of the figure. The figure shows different curves for the several p_1 values. Each point on each curve averages 20 planning runs. The curves of $p_1 = \{0.01, 0.05\}$ were left out for clarity.

We first compare the estimation-indifferent approach with ACE. Consider the worst-case where the cost of *all* actions is estimated ($p_1 = 1$, solid purple), and the target bound requires maximum accuracy ($\mathcal{B} = 1$). Here, ACE utilizes only 62% of the expensive estimators. For lower p_1, it improves up to 46% ($p_1 = 0.1$). This is due to the condition on $\underline{g}$ in line 14 of ACE, which avoids unnecessary estimations in case another path is examined that leads to a node that is already in OPEN and has lower f-value. Without sacrificing accuracy, ACE is superior to its basic competitor that uses full estimation (indicated as the straight line of ratio 1).

Next, the generally decreasing trend of the curves in Fig. 1 suggests that substantial savings may be achieved even by minor relaxation of the uncertainty bound (allowing greater $\mathcal{B}$). Furthermore, as p_1 decreases the curves drop rapidly, allowing ACE to further increase its efficiency.

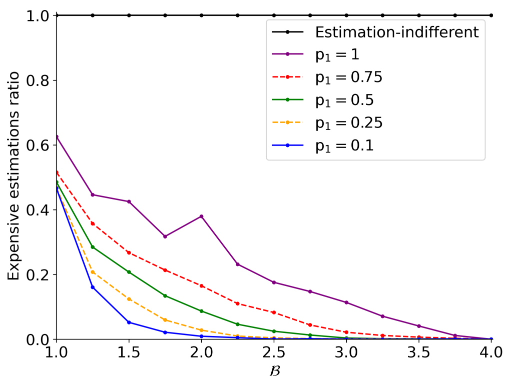

Figure 1: Normalized expensive estimations versus target bound $\mathcal{B}$. The curves from the bottom (blue, solid), to the top (purple, solid) correspond to the p_1 values $0.1, 0.25, 0.5, 0.75$ and 1. Success rate is 100% for all runs. The uppermost curve (black, solid) is the baseline estimation-indifferent

Run-Time and Quality. Figure 2 shows actual and approximate planning run-time (measured in CPU seconds), on the vertical axis (logarithmic scale). As the time taken by expensive estimators is arbitrary in these experiments, we wanted to get a feel for the planning run-time under various assumptions of computation time. The bottom curve in the figure is the actual run-time of PlanDEM without the estimation. The other curves, bottom-to-top, show what the run-time would be had we added the estimation run-time (number of expensive estimates multiplied by assumed per-estimate run-time). This emphasizes the importance of avoiding unnecessary estimations, which dominate the total run-time when estimation time is high (compared to the heuristic estimation time, contained within the baseline). It also strengthens the argument mentioned above that significant potential savings can be obtained by even slightly loosening the uncertainty bound.

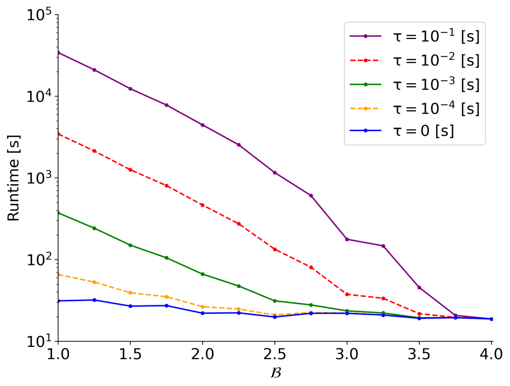

Figure 2: PlanDEM run-time in CPU seconds. Bottom (blue, solid) to top (purple, solid) curves correspond to per-estimation times of $0, 10^{-4}, 10^{-3}, 10^{-2}$ and 10^{-1} seconds for the expensive estimators. $p_1 = 0.5$ for all cases.

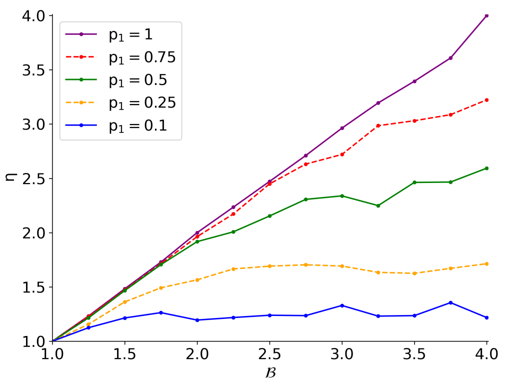

Figure 3: Mean η vs $\mathcal{B}$ target bounds. The curves bottom (blue, solid) to top (purple, solid) correspond to the p_1 values $0.1, 0.25, 0.5, 0.75$ and 1.

We now move to examine the quality of the solutions produced by ACE. Figure 3 shows mean η values achieved for different target $\mathcal{B}$ bounds. Recall that we consider shorter planning time to be better than achieving higher accuracy. Hence, the fact that η tends to $\mathcal{B}$ when p_1 is high indicates that ACE is efficient w.r.t. usage of expensive estimations. When p_1 values are low, most costs are precisely known, and overall uncertainty should be low even with a small amount of expensive estimations (see Fig. 1). Indeed, η values are well below the target bounds.

Evaluation of ESE

The section above examines ACE when it can always succeed in finding a $\mathcal{B}$-solution, and thus ESE is never invoked. This section examines cases where it might not. We introduce probabilities for the existence of the second and third estimators, denoted by p_2 and p_3. When setting either p_2 or p_3 to values different than 1, the achievable η increases, as tighter estimates are no longer available. This creates cases where ACE can fail, and ESE might be applicable, and will be invoked. For our experiment we used $p_1, p_2 \in \{0.25, 0.5, 0.75, 1\}$, $p_3 \in \{0.25, 0.5, 0.75\}$ and $\mathcal{B} \in \{1.5, 2, 2.5, 3, 3.5\}$. This resulted in 4800 planning problem variants to be solved.

The results are summarized in Table 1. Each row corresponds to a different $\mathcal{B}$ target, and summarizes 960 problems. Left to right, the columns show the number of problems (in parentheses, percentage of problems) in which ACE succeeded, ESE was applicable and invoked, ESE was successful, the resulting η and usage of costly estimations for both ACE and ESE. The table points to several insights.

First, the number of ESE invocations is not equal to the number of ACE failures. There are cases where ACE exhausted all estimation options for the actions in the plan and thus ESE is not applicable. As the success ratio of ACE increases with $\mathcal{B}$, the number of ESE invocations decreases (as this procedure is required less frequently).

Second, a key insight is that ESE has a considerable success ratio, which increases with $\mathcal{B}$. This is striking, given that it utilizes only a small amount of estimates w.r.t. the amount ACE uses, i.e., its affect on the total run-time is minor.

To explain this result, we examine the change in η due to ESE. The data in column 5 from the table indicates values averaged over both successful and failed runs (when the bound is not met), without normalization. We therefore examine the normalized relative change in η, defined as

$$\eta_{rel} := (\eta^{\mathsf{ESE}} - \eta^{\mathsf{ACE}})/(\eta^{\mathsf{ACE}} - 1),$$

where the value 1 in the denominator, which is the best (lowest) achievable η, serves as to normalize the change.

The mean normalized relative change is only -5.72%. This implies that ACE typically comes very close to the target (up to roughly several percent). It then becomes relatively easy for ESE to use a few more estimates, to drive η below the target.

An ACE Experiment on Real-World Data

Inspired by the contrast between the simplistic estimates of the Transport Sequential domain and the commercial-grade estimation carried out by TruckNet ((TruckNet 2021); see introduction), we created a package delivery problem based on real data for the Ontario province in Canada. The problem is deceptively small: 9 cities, with 30 road segments (distances from Google Maps), 2 trucks each with 4-package capacity, and 6 packages to be delivered (spread across different initial and terminal locations). There are three actions: drive, pickup and drop. The objective is to minimize monetary costs (a function of both distance and time), subject to the inherent uncertainty of travel times.

The cost per distance was computed online based on the truck model depreciation, financing, licensing, insurance, and fuel prices for the region. The cost per minute was computed from reported driver salaries in Canada. Two estimators were used for the drive action cost: the first (cheap) assumed the speed would be between heavy traffic (20km/h) and no traffic (100km/h) conditions; the second (more costly) estimator used pessimistic and optimistic times reported by Google Maps for a specific day and hour. Both estimators yield travel time in minutes, which was then used with the cost-per-minute and cost-per-distance (full details appear in the appendix). In order to test a more computationally intensive scenario, we assumed that the costs are state-dependent, which significantly increases the required time for overall estimation.

Using ACE saved between 60% to 100% of the need to use the costly estimator, compared to estimation-indifferent planning. The success rate was 100% on feasible $\mathcal{B}$ targets, and in the cases of unfeasible targets (too demanding $\mathcal{B}$ values) there were no unused estimators for the plan found, so ESE was not used. The best attainable estimates had uncertainty ratios ranging from 8.5% to 53.8%, demonstrating that cost uncertainty is often unavoidable. In such cases, ACE is superior to standard planning. Assuming a realistic 10ms online-query time for each invocation of the costly second estimator, and even assuming only 60% savings (worst case found) this translates to a planning time of 7 hours using ACE, instead of 18.

$\mathcal{B}$	ACE Success (%)	ESE Invoked (%)	ESE Success (%)	η ACE (ESE)	Costly Estimations ACE (ESE)
1.5	276 (28.75)	174 (18.13)	50 (28.74)	2.23 (1.98)	132566045 (511)
2	548 (57.08)	171 (17.81)	59 (34.50)	2.63 (2.14)	145826527 (602)
2.5	701 (73.02)	145 (15.10)	52 (35.86)	2.95 (2.33)	136260264 (544)
3	806 (83.96)	97 (10.10)	55 (56.70)	3.29 (2.51)	93636149 (382)
3.5	897 (93.44)	38 (3.96)	23 (60.53)	3.67 (2.68)	29996783 (144)

Table 1: Summarized data of ACE & ESE experiments (960 runs per $\mathcal{B}$ value). The values in column 5 are averages of η over all instances where ESE took place, and the values in column 6 are sums of costly estimations over all those instances.

Related Work

ACE's repeated evaluation and selection between cost estimators *complements* the process of choosing between multiple heuristics during planning (Karpas et al. 2018). Heuristics are applied to states (vertices in the state-space graph), while costs apply to actions (edges).

In shortest-path search, cost uncertainty is considered by assuming explicit or implicit knowledge about edge cost distributions, and then solved by performing calculations involving the full graph, typically minimizing expectation (Kwon, Lee, and Berglund 2013; Shahabi, Unnikrishnan, and Boyles 2015). ACE makes no such assumptions.

Related efforts tackle graph search in the context of motion planning, where verification of edge existence is expensive (Narayanan and Likhachev 2017; Mandalika et al. 2019). The solution approach dynamically decides when to search and when to evaluate edges, so as to minimize overall planning time, but is limited to existence verification. This is a special case of general action cost computation in P-MACE (when the cost tends to infinity, an edge can be considered non-existent). Existence verification is carried out only once per selected edge, whereas P-MACE may require multiple estimations for the same edge.

A closer line of work addresses shortest path problems where obtaining the true edge cost is considered to be computationally expensive. Dellin and Srinivasa (2016) present a framework for lazy search w.r.t. expensive edge evaluations. They assume two estimates are provided for each edge: a computationally-cheap lower-bound (no upper bound), and a computationally-expensive exact cost. Clearly this is a special class of P-MACE problems. The lazy search cannot solve P-MACE problems in general, because it does not admit upper bound estimates, and cannot target $\mathcal{B}$-bounded solutions. Technically, the lazy search algorithm strictly requires two estimates, of which one is the exact cost. This requirement does not exist for ACE.

We emphasize that P-MACE problems can, in general, be solved *without* utilizing all the most expensive estimators in the solution path, hence it is insufficient to simply be lazy w.r.t. to the most expensive estimators.. For instance, ACE can sometimes solve a P-MACE problem while utilizing *all* estimates for some edges in the solution path, some of the estimates for other edges in the path, and only one estimate for the remaining edges.

Discussion

ACE's theoretical properties hold regardless of the order of the estimators given by GetEstimator. However, even *partial ordering* of the estimators by expected run-time can be used by GetEstimator to return cheaper estimators first, making ACE faster in practice.

Typically, algorithms that try to reduce resource consumption of one type, have to make the sacrifice of consuming more resources of a different type (the classical trade-off being run-time vs. memory). ACE is able to greatly reduce the number of expensive estimators utilized, compared to the estimation-indifferent approach (as demonstrated in Fig. 1). Since the estimation-indifferent baseline considered here has the same node-expansion behavior of standard A^* (when it considers only the tightest lower bounds as costs), it is guaranteed to be optimally efficient (when used with a consistent heuristic).

One may therefore expect that ACE should be inefficient w.r.t. the number of node expansions. But this is not true. In fact, experiments suggest that ACE uses roughly the same number of node expansions as the estimation-indifferent baseline. We believe the trade-off is elsewhere. ACE uses a *best-effort approach*, and so it is not guaranteed to return a $\mathcal{B}$-optimal plan in the general case, but only in the special case where an exact-cost estimate is given for every action. We expect that future extension of ACE that guarantee general $\mathcal{B}$-optimality will need to increase search effort (i.e., will expand a larger number of nodes).

An assumption made in this paper is that estimators represent cost uncertainty by an interval composed of lower and upper bounds. In cases where statistical estimators are used, statistical properties (e.g., standard deviation) may used instead as bounds, in which case the plan accuracy bound should be considered soft (or approximate) rather than strict.

Conclusions

We introduced *planning with multiple action cost estimators* (P-MACE), a generalization of deterministic planning that allows the domain expert to provide the planner with multiple cost estimators, each with its own uncertainty and computational requirements. During planning, the planner is then responsible for balancing the computational cost of estimating action costs, and the accuracy of the estimates. We then introduced ACE—a generalization of A^*—and showed that it is optimal in case each action cost can ultimately be estimated exactly (and also in other cases). To improve upon it when it fails to meet the target, we introduced a post-search procedure that increases ACE's success ratio. Extensive experiments show the algorithms are very effective.

ACE is the first attempt to solve P-MACE problems as introduced here. We plan to focus future algorithmic improvements on general $\mathcal{B}$-optimality. Another important future direction is to improve the selection of estimators so as to minimize their use, by considering meta-information, such as predicted values for c_{min}, c_{max} and/or run-time prior to the actual application of an estimator.

Appendix: Experiment Details and Results

Domains and Problems Used in the Experiments

For the empirical evaluation based on IPC problem domains, we used domains and problems that appeared in the international planning competitions of 2008, 2011, 2014 and 2018. For each domain we chose two problems (ranging from small to large scale), and for each of them we tested all configurations reported in the paper (resulting in many different variants per original problem file). The full list of domains (problems) follows below, and the files can be retrieved from (Seipp et al. 2016).

- elevators-opt08-strips (p04, p06)
- barman-opt11-strips (pfile01-003, pfile01-004)
- floortile-opt11-strips (opt-p05-009, opt-p06-011)
- sokoban-opt11-strips (p04, p07)
- transport-opt11-strips (p02, p04)
- woodworking-opt11-strips (p06, p12)
- tetris-opt14-strips (p03-4, p04-6)
- agricola-opt18-strips (p08, p10)
- caldera-split-opt18-adl (p05, p10)
- data-network-opt18-strips (p17, p20)

Costs and Estimators

We generally used the original cost depicted in the PDDL domain and problem files, denoted as $c_{PDDL}(e)$. However, PlanDEM supports solely positive integer action costs, as it extends FD and thus inherits its main software architecture and data structures. As a result, for problems that have $c(e) = 1$, it is not possible to bound an estimate from below. In these cases we changed the original cost so as to be able to experiment with estimators that provide a bound lower than the original cost.

Thus, in order to obtain the desired uncertainty ratios for each edge e, we used $c_{PDDL}(e)$ as a basis for the following transformation.

- If the edge cost was to be estimated (i.e., with probability p_1 as described in the experiment section of the paper), then the list of estimators is given by $\Theta(e) = \{\theta_1^e, \theta_2^e, \theta_3^e\}$, where the estimators respectively return
  $$c_{min}^1(e) = 1 \times c_{PDDL}(e), c_{max}^1(e) = 4 \times c_{PDDL}(e),$$
  $$c_{min}^2(e) = 2 \times c_{PDDL}(e), c_{max}^2(e) = 4 \times c_{PDDL}(e),$$
  $$c_{min}^3(e) = 2 \times c_{PDDL}(e), c_{max}^3(e) = 2 \times c_{PDDL}(e).$$
  In this case, the true cost is taken to be $c(e) = 2 \times c_{PDDL}(e)$.
- Otherwise (true cost known and not estimated): $\Theta(e) = \{\theta_1^e\}$, where θ_1^e returns
  $$c_{max}^1(e) = c_{min}^1(e) = c_{PDDL}(e).$$
  Hence, in this case $c(e) = c_{PDDL}(e)$.

Ontario Transportation Logistics Example

We describe the details of the experiment with real-world data. The road system considers 9 cities with 30 road segments connecting them, where each road length is specified using a 100 meter resolution, where the data is taken from Google Maps. The cost function per action a depends on the distance traveled d and on time duration t, and is given by $c(a) = c_1 \times d + c_2 \times t$, where $c_1 = 0.56$ [USD/km] and $c_2 = 0.5$ [USD/minute] are the cost-per-km and cost-per-minute coefficients. c_1 was taken from (Vincentric 2022) and is based on data tailored to a specific set of assumptions: Ford Transit 150, 2021 model, with base crew medium sized roof slide 148WB AWD trim, commuting 30000 km annually, in Ontario province. It calculates an average total cost-per-km based on the above specifications, considering current fuel prices, depreciation and maintenance costs, license and registration fees, insurance costs and even financing. c_2 is based on an estimate of a truck driver employer's cost in Canada (see e.g., data from Glassdor).

The domain file is similar to the aforementioned "Transport Sequential", having three action templates: drive, pick-up and drop, where the prices of the two latter actions are assumed to be exact and are based on a 20 minutes time duration. Each drive action has two estimators: the first, which doesn't incur additional run-time, assumes nothing about the specific road segment and thus uses conservative estimates of 20 km/hour and 100 km/hour for unfavorable and favorable road conditions, respectively; and the second uses pessimistic and optimistic travel time estimates taken from Google Maps by manual queries for a specific day and hour (Friday, 10 AM) and the fastest route. All the required data for reproducing the experiment is contained in 3 files, corresponding to the domain, problem and estimators, that appear in (Weiss and Kaminka 2023).

We evaluated ACE on the problem for $\mathcal{B} \in \{1, 1.1..., 2.4\}$, and the results are summarized in Table 2. As can be seen, the results are in full correspondence with those obtained by experimenting with the modified IPC benchmarks.

We highlight several interesting conclusions from the experiment:

- ACE saved between 60% and $\sim$100% costly estimations compared to estimation-indifferent planning, while maintaining a 100% success rate on feasible requirements (as $\mathcal{B} = 1, 1.1$ turned out to be out of reach).
- The best attainable estimates had uncertainty ratios ranging from 8.5% to 53.8%, demonstrating that cost uncertainty is oftentimes unavoidable. In these cases standard planning would generate inferior plans.
- Assuming a 1ms query time for travel time prediction (representing access time to a local database), and even in the worst result of only 60% savings, planning time is 43 minutes instead of 107 minutes for estimation-indifferent planning. For query times of 10ms (representing a very fast access to an online database), this translates to 7 hours of planning time instead of 18 hours.

We point out that this experiment was based on a pre-compiled file containing the required estimation data, so

$\mathcal{B}$	I_s	η	θ_2 Used	θ_2 Saved	θ_2 Usage (%)	T_0[s]	T_1[s]	Saved T_1[s]	T_{10}[s]	Saved T_{10}[s]
1	0	1.18	2537169	3858085	39.67	54.12	2591.29	3858.09	25425.81	38580.85
1.1	0	1.18	2537169	3858085	39.67	61.89	2599.06	3858.09	25433.58	38580.85
1.2	1	1.18	2565641	4254910	37.62	66.16	2631.80	4254.91	25722.57	42549.10
1.3	1	1.28	2653524	7091775	27.23	93.93	2747.46	7091.78	26629.17	70917.75
1.4	1	1.39	2469987	8222062	23.10	101.26	2571.25	8222.06	24801.13	82220.62
1.5	1	1.47	2253078	8587468	20.78	102.10	2355.18	8587.47	22632.88	85874.68
1.6	1	1.58	2047551	8909325	18.69	106.90	2154.45	8909.33	20582.41	89093.25
1.7	1	1.66	1809278	8843618	16.98	105.22	1914.50	8843.62	18198.00	88436.18
1.8	1	1.76	1508847	9375424	13.86	94.54	1603.39	9375.42	15183.01	93754.24
1.9	1	1.88	1149769	8486273	11.93	72.78	1222.55	8486.27	11570.47	84862.73
2	1	1.96	722353	8102761	8.19	84.37	806.72	8102.76	7307.90	81027.61
2.1	1	2.03	237430	6863791	3.34	69.62	307.05	6863.79	2443.92	68637.91
2.2	1	2.05	733	6349392	0.01	60.23	60.96	6349.39	67.56	63493.92
2.3	1	2.16	31	6350040	5×10^{-4}	61.55	61.58	6350.04	61.86	63500.40
2.4	1	2.24	7	6350479	10^{-4}	60.59	60.59	6350.48	60.66	63504.79

Table 2: Summarized data of ACE experiments with the Ontario Transportation Logistics problem. Notations: I_s is a success indicator, namely $I_s = 1$ if $\eta \leq \mathcal{B}$ and 0 otherwise, θ_2 denotes the costly estimations, and T_0, T_1, T_{10} correspond to planning times using $\tau_2 = 0, 1, 10$ ms, respectively. Usage and savings are w.r.t. to the results of estimation-indifferent planning.

that during planning cost estimates were obtained by repeated access to the file, with many ground actions having the same cost estimates. In such cases where multiple actions (or edges) have the same estimate, using a cache mechanism, in order to reduce calls to external modules, should significantly improve the results. The design and implementation of an effective cache is left for future research.

Appendix: Proofs of Theoretical Results

In order to prove Lemma 1 we require another auxiliary result, which demonstrates that heuristic consistency is preserved when using lower bounds of edge costs for its calculation (instead of the true edge costs). Note that often the term heuristic function, in the context of graph search (or AI planning), is an abuse of notation used to refer to a more general computational procedure, that is parameterized by the edge costs of the graph (or by the action costs in the planning problem), so that only after fixing the costs, a standard heuristic is obtained. Further note that it is typical to attribute a theoretical property, such as consistency, to such a computational procedure, if the property holds for any heuristic obtained from the procedure after fixing the costs, regardless of their specific values. For stating our result, we need to differentiate between the two. Hence, we call the more general computational procedure a parameterized heuristic and denote it as usual by h, and we denote a standard heuristic obtained by it using a subscript, e.g., h_c, where c is a cost function defining the costs of the problem.

Lemma 2 (Consistency Preservation)**.** *Given a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and two edge cost functions $\alpha : \mathcal{E} \to [0, \infty)$, $\beta : \mathcal{E} \to [0, \infty)$ that satisfy $\alpha(e) \leq \beta(e)$ for every edge $e \in \mathcal{E}$, we obtain the weighted digraphs $\mathcal{G}_\alpha = (\mathcal{V}, \mathcal{E}, \alpha)$, $\mathcal{G}_\beta = (\mathcal{V}, \mathcal{E}, \beta)$. If h is a consistent parameterized heuristic, then h_α is consistent w.r.t. $\mathcal{G}_\beta$.*

Proof of Lemma 2. Due to the fact that h is a consistent parameterized heuristic, it immediately follows that h_α is consistent w.r.t. $\mathcal{G}_\alpha$. From the definition of consistency, this means that $h_\alpha(n) \leq \alpha((n, s)) + h_\alpha(s)$ and $h_\alpha(G) = 0$ is satisfied for every node n and every descendant s of n, and every goal node G in $\mathcal{G}_\alpha$. Using $\alpha(e) \leq \beta(e)$ we obtain $h_\alpha(n) \leq \beta((n, s)) + h_\alpha(s)$, where again, this holds for every node n and every descendant s of n. Additionally, every goal node G in $\mathcal{G}_\alpha$ is also a goal node in $\mathcal{G}_\beta$, so $h_\alpha(G) = 0$ is also satisfied for every goal node in $\mathcal{G}_\beta$. Thus, h_α satisfies the definition of consistency w.r.t. $\mathcal{G}_\beta$. $\square$

We can now prove Lemma 1.

Proof of Lemma 1. As mentioned before, ACE uses heuristic values that are computed by h (which is a consistent parameterized heuristic) and based on the lower bound estimate given for each relevant edge e by the first estimator returned by GetEstimator(e). We obtain a (standard) consistent heuristic h_α, with α defined to be an edge cost function, where the cost of each edge e is taken to be the lower bound estimate of the first estimator returned by GetEstimator(e). During the search ACE potentially utilizes additional estimations (as it tries to meet the target $\mathcal{B}$). Denote by $\Phi \subseteq \Theta_\Sigma$ the set that includes exclusively all estimators invoked by ACE during the search, and further denote β as an edge cost function, where the cost of each edge e is taken to be the tightest (highest) lower bound estimate of all the estimators returned by GetEstimator(e) during the search. Since $\alpha(e) \leq \beta(e)$ for every edge e, Lemma 2 assures that h_α is consistent w.r.t. $\mathcal{G}_\beta$ (where its nodes and edges are defined by the digraph that corresponds to the planning problem $\mathcal{P}$). Then, following the main proof arguments of A^*'s optimality, consistency of h_α implies non-decreasing f-values along any path, thus whenever ACE selects a node for expansion, an optimal path (w.r.t. β) to that node has been found. Hence, the first plan π found by ACE has plan lower bound $c^\Phi_{min}(\pi)$ equal to the optimal plan lower bound w.r.t. Φ, i.e., $c^\Phi_{min}(\pi) = c^*_\Phi$, implying that π is an optimal optimistic plan w.r.t. Φ. $\square$

Acknowledgements

The authors thank Alexander Shleyfman for his constructive feedback. The research was partially funded by ISF Grant #2306/18 and BSF-NSF grant 2017764. Thanks to K. Ushi. Eyal Weiss is supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities and by Bar-Ilan University's President Scholarship.

References

Allen, C.; Katz, M.; Klinger, T.; Konidaris, G.; Riemer, M.; and Tesauro, G. 2021. Efficient Black-Box Planning Using Macro-Actions with Focused Effects. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*.

Bonet, B.; and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence*, 129(1-2): 5–33.

Coles, A.; Coles, A.; Olaya, A. G.; Jiménez, S.; López, C. L.; Sanner, S.; and Yoon, S. 2012. A survey of the seventh international planning competition. *AI Magazine*, 33(1): 83–88.

Dellin, C.; and Srinivasa, S. 2016. A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 26, 459–467.

Dornhege, C.; Eyerich, P.; Keller, T.; Trüg, S.; Brenner, M.; and Nebel, B. 2009. Semantic attachments for domain-independent planning systems. In *Nineteenth International Conference on Automated Planning and Scheduling*.

Fox, M.; and Long, D. 2003. PDDL 2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20: 61–124.

Francès, G.; Ramiŕez, M.; Lipovetzky, N.; and Geffner, H. 2017. Purely Declarative Action Descriptions are Overrated: Classical Planning with Simulators. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*.

Garrett, C. R.; Chitnis, R.; Holladay, R.; Kim, B.; Silver, T.; Kaelbling, L. P.; and Lozano-Pérez, T. 2021. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4: 265–293.

Ghallab, M.; Nau, D.; and Traverso, P. 2016. *Automated planning and acting*. Cambridge University Press.

Gregory, P.; Long, D.; Fox, M.; and Beck, J. C. 2012. Planning Modulo Theories: Extending the Planning Paradigm. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2): 100–107.

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

Karpas, E.; Betzalel, O.; Shimony, S. E.; Tolpin, D.; and Felner, A. 2018. Rational deployment of multiple heuristics in optimal state-space search. *Artificial Intelligence*, 256: 181–210.

Koenig, S.; Likhachev, M.; and Furcy, D. 2004. Lifelong Planning A^*. *Artificial Intelligence*, 155(1–2): 93–146.

Kwon, C.; Lee, T.; and Berglund, P. 2013. Robust shortest path problems with two uncertain multiplicative cost coefficients. *Naval Research Logistics (NRL)*, 60(5): 375–394.

LaValle, S. M. 2006. *Planning Algorithms*. Cambridge Univerity Press.

Mandalika, A.; Choudhury, S.; Salzman, O.; and Srinivasa, S. 2019. Generalized lazy search for robot motion planning: Interleaving search and edge evaluation via event-based toggles. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 745–753.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL-the planning domain definition language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

Narayanan, V.; and Likhachev, M. 2017. Heuristic search on graphs with existence priors for expensive-to-evaluate edges. In *Twenty-Seventh International Conference on Automated Planning and Scheduling*.

Seipp, J.; Pommerening, F.; Sievers, S.; and Francès, G. 2016. Fast Downward benchmark collection. https://github.com/aibasel/downward-benchmarks. Accessed: 2023-03-14.

Shahabi, M.; Unnikrishnan, A.; and Boyles, S. D. 2015. Robust optimization strategy for the shortest path problem under uncertain link travel cost distribution. *Computer-Aided Civil and Infrastructure Engineering*, 30(6): 433–448.

Stentz, A. 1994. Optimal and Efficient Path Planning for Partially-Known Environments. In *Proceedings of IEEE International Conference on Robotics and Automation*, 3310–3317.

TruckNet. 2021. Trucknet Enterprise: Revolutionising the World of Freight Management. Logistics Tech Outlooks: https://www.trucknet.io/wp-content/uploads/2021/10/629764_418641774086freight-management-europe-cover774086418641.pdf. Accessed: 2021-12-15.

Vincentric. 2022. Driving Costs Calculator. https://carcosts.caa.ca/. Accessed: 2023-03-14.

Weiss, E. 2022. A Generalization of Automated Planning Using Dynamically Estimated Action Models–Dissertation Abstract. In *32nd International Conference on Automated Planning and Scheduling*, 1–3.

Weiss, E.; and Kaminka, G. A. 2022. Position Paper: Online Modeling for Offline Planning. In *Proceedings of the 1st ICAPS Workshop on Reliable Data-Driven Planning and Scheduling*.

Weiss, E.; and Kaminka, G. A. 2023. PlanDEM. https://github.com/eyal-weiss/plandem-public. Accessed: 2023-03-14.