A Fast Heuristic Search Approach for Energy-Optimal Profile Routing for Electric Vehicles

Saman Ahmadi, Mahdi Jalili

School of Engineering, RMIT University, Australia
saman.ahmadi@rmit.edu.au, mahdi.jalili@rmit.edu.au

Abstract

We study the energy-optimal shortest path problem for electric vehicles (EVs) in large-scale road networks, where recuperated energy along downhill segments introduces negative energy costs. While traditional point-to-point pathfinding algorithms for EVs assume a known initial energy level, many real-world scenarios involving uncertainty in available energy require planning optimal paths for all possible initial energy levels, a task known as energy-optimal profile search. Existing solutions typically rely on specialized profile-merging procedures within a label-correcting framework that results in searching over complex profiles. In this paper, we propose a simple yet effective label-setting approach based on multi-objective A* search, which employs a novel profile dominance rule to avoid generating and handling complex profiles. We develop four variants of our method and evaluate them on real-world road networks enriched with realistic energy consumption data. Experimental results demonstrate that our energy profile A* search achieves performance comparable to energy-optimal A* with a known initial energy level.

Introduction

Optimal pathfinding is a well-known and classic problem in AI: determining the least-cost path from a starting point to a target location within a network. For networks with non-negative edge costs, this problem can be efficiently solved using dynamic programming techniques such as Dijkstra's algorithm (Dijkstra 1959), or more efficient heuristic-guided methods like A* search (Hart, Nilsson, and Raphael 1968). However, the problem becomes more challenging in the presence of negative edge costs. In this case, traditional algorithms like Bellman-Ford (Bellman 1958; Ford Jr 1956) are applicable but tend to be slower than their counterparts designed for graphs with non-negative costs. Nonetheless, if the graph is free of negative cycles, more efficient techniques such as Johnson's algorithm (Johnson 1977) can be used. These methods achieve performance comparable to Dijkstra's algorithm, albeit after a reweighting step. A practical application of optimal pathfinding with negative edge costs arises when energy is the primary cost metric, as it can be both consumed (positive cost) and regenerated (negative cost) along an edge. Due to the law of conservation of energy, such graphs are typically free of negative cycles, making reweighting techniques viable in practice.

One key example of optimal pathfinding with energy costs is path planning for electric vehicles (EVs) in transportation networks. Driven by the rapid advancement of vehicle and battery technologies, EVs have permeated nearly every form of transportation, including (but not limited to) daily commuting with light EVs, public transit with electric buses, and package delivery with e-bikes. In the context of pathfinding, EVs exemplify systems with negative energy costs, as they feature energy recuperation (e.g., via regenerative braking), which can result in negative energy costs on certain road segments. Given that the energy available to an EV is constrained by its battery capacity, reaching certain destinations (including charging stations) may only be possible via an energy-optimal path, i.e., a path from origin to destination with the lowest possible energy requirement. Energy-optimal paths are critical as they determine feasibility: an EV cannot reach its destination if no minimum-energy path exists for the available energy at the origin. It is therefore essential to have algorithms capable of reasoning about and efficiently computing energy-optimal paths.

When the EV's energy at the starting point is known, energy-optimal pathfinding can be formulated as a single-criteria pathfinding problem, where the available energy at any node along the energy-optimum path to the destination can be computed through simple algorithmic modifications to ensure it remains within the range. The work in Artmeier et al. (2010) represents one of the first attempts to solve this problem efficiently by such modification through two approaches: an adapted version of the Bellman-Ford algorithm (with polynomial time complexity), and a modified version of Dijkstra's algorithm that allows vertex re-exploration (with exponential time complexity). This research was later extended in Sachenbacher et al. (2011), where the authors applied Johnson's reweighting technique (Johnson 1977) and an adapted A*-based algorithm to further reduce computational complexity to quadratic time. A key component of their algorithms is a potential energy function that enables the upshifting of edge costs into non-negative reduced costs, and also serves as a lower-bound heuristic in the A* framework. Inspired by this idea, subsequent research has focused on preprocessing-based approaches to accelerate energy-optimum path queries. These methods include ap-

plying Johnson's edge-cost shifting technique to precompute a graph with only non-negative edge costs, as well as hierarchical speed-up techniques such as Contraction Hierarchies (CH) (Geisberger et al. 2008). In particular, the work in Baum et al. (2013, 2019) continues this line of research, exploring various potential functions for cost shifting and evaluating their integration with speed-up techniques such as CH (Delling et al. 2017; Jung and Pramanik 2002).

A more general case of energy-optimal pathfinding arises when the initial energy level is unknown. This critical scenario appears in many real-world applications, such as planning multi-leg trips in EV routing, where the initial energy level of each leg depends on the final state of charge (SoC) from the previous trip (Ahmadi et al. 2021), or in bidirectional settings (with a backward profile search from destination to origin), where the energy level at the destination cannot be predetermined (Baum et al. 2020). In this setting, cost-optimal paths must be computed for all possible energy values, ranging from zero to EV's maximum battery capacity. This task is known as *energy profile search*. Surprisingly, energy-optimal profile search for EVs remains under-explored. The first attempts to address this problem were made by Baum et al. (2013) and Schönfelder, Leucker, and Walther (2014), where Dijkstra's Algorithm and A* search were adapted to handle energy profiles. Both approaches are based on a label-correcting strategy that incrementally links and merges energy profiles during the search. However, these methods often require complex rules to extend and relax energy profiles, adding significant overhead due to the inefficiency of explicitly merging profiles across the entire SoC range to maintain only one profile per graph vertex. Continuing along this line, Baum et al. (2020) extended their earlier work within a larger path planning context with charging stations, but still relied on the conventional label-correcting approach to handle uncertainty in initial SoC. Despite recent progress in accelerating energy-optimal pathfinding queries with (preprocessing-based) speed-up techniques, the core problem of energy profile search remains largely unaddressed: existing profile search algorithms are still complex, often requiring specialized procedures and implementation tricks to enable the propagation of profiles during the search. This hinders their practical applicability and adoption.

Contributions: This paper presents a novel yet simple label-setting approach for energy-optimal profile search for EVs. The method adopts a multi-objective search strategy to propagate profiles in a best-first manner while incorporating efficient pruning mechanisms to discard unpromising profiles and to avoid expensive profile merging often observed in label-correcting schemes. Experiment results on realistic instances demonstrate that our proposed approach offers a fast and practical framework for addressing energy-optimal profile queries in large road networks.

Problem Definition and Background

Consider a road network represented as a directed graph $G = (S, E)$, where S is a finite set of states (intersection) and $E \subseteq S \times S$ is a set of edges (road segment). Each edge has a real-valued energy cost, denoting the amount of energy required to traverse the link. Edge cost may be negative due to energy recuperation, and can be retrieved via the function $cost : E \to \mathbb{R}$. A path π in G is defined as an ordered sequence of n states $\{u_1, \ldots, u_n\}$ where $(u_i, u_{i+1}) \in E$ for $i \in \{1, \ldots, n-1\}$. Now, consider an EV starting with an initial energy of $\mathcal{E}_{init} \in \mathbb{R}^+$ at an initial location $start \in S$, and a maximum energy limit of $\mathcal{E}_{max} \in \mathbb{R}^+$ (battery capacity). The objective of energy-optimal pathfinding is to determine an optimum path π^* that allows the EV to travel from $start$ to a target location $goal \in S$ with the least amount of energy consumed, while ensuring that the energy level of the EV remains within the $(0, \mathcal{E}_{max}]$ range at all locations along π^*. If $\mathcal{E}_{\text{init}}$ is unknown, the objective becomes identifying, for each possible value of $\mathcal{E}_{\text{init}}$ within the allowable range, the corresponding energy-optimal path, i.e., computing all minimum-energy paths as a function of initial energy $\mathcal{E}_{\text{init}}$.

Algorithm 1: Energy-optimal A* Search

Inputs: A problem instance (G, $cost$, h, $start$, $goal$, $\mathcal{E}_{init}$)
Output: Energy requirement of the optimal path

```
Open ← ∅, C(u) ← ∞ for all u ∈ S
x ← new node with s(x) = start
g(x) ← 0, f(x) ← h(start), C(start) ← 0, P(x) ← ∅
Add x to Open
while Open ≠ ∅ do
    Extract from Open node x with the smallest f-value
    if g(x) > C(s(x)) then continue
    if s(x) = goal then break
    for each v ∈ Succ(s(x)) do
        y ← new node with s(y) = v
        g(y) ← g(x) + cost(s(x), v)
        f(y) ← g(y) + h(v)
        P(y) ← x
        if g(y) > E_init or f(y) > E_init then continue
        if E_init − g(y) > E_max then
            g(y) ← E_init − E_max
        if g(y) < C(v) then
            C(v) ← g(y)
            Add y to Open
return C(goal)
```

Energy-optimal pathfinding with A*

We now briefly review energy-optimal pathfinding on the basis of A*. Following the heuristic search literature, we define partial paths to states as search nodes. More precisely, a node x represents a path originating from $start$. Each node x is associated with a state $s(x)$ (i.e., the last state in the partial path), a cost value $g(x)$ representing the cumulative energy consumed from $start$ to $s(x)$, a reference $\mathcal{P}(x)$ pointing to its parent node, and a total cost estimate $f(x)$, a lower bound on energy cost from $start$ to $goal$ via x.

The search in A* is guided by nodes with the smallest cost estimates, or f-values, which are conventionally computed using a consistent and admissible heuristic function $h : S \to \mathbb{R}$ such that $f(x) = g(x) + h(s(x))$ (Hart, Nilsson, and Raphael 1968). h is consistent if $h(u) \leq h(v) + cost(u, v)$ for every edge $(u, v) \in E$, and also admissible if we ad-

ditionally have $h(goal) = 0$. Consistent heuristics offer a key advantage in negative-weight graphs: f-values in A* are still guaranteed to increase monotonically during the search (Ahmadi et al. 2024a). This property inherently eliminates the need for reweighting techniques and provides a robust framework for solving the problem in dynamic settings, where both energy models and heuristic functions may change between queries (Ahmadi, Raith, and Jalili 2025). However, this benefit comes at an additional computational cost, as some better informed heuristic functions may involve non-linear and potentially expensive computations.

As discussed in Schönfelder, Leucker, and Walther (2014), consistent energy heuristic functions for EVs can be naturally derived using physical energy models, typically formulated as lower bounds on the sum of kinetic and potential energy. In this context, the kinetic energy estimate is modeled as a function of air-line distance, while the potential energy component is based on elevation differences. See Ahmadi and Jalili (2025) for two such heuristic functions.

Algorithm 1 presents a pseudocode for energy-optimal A* search, adapted from Schönfelder, Leucker, and Walther (2014). It begins by initializing a node with the $start$ state and inserting it into $Open$, a priority queue that stores all generated but unexplored nodes. The algorithm also initializes a scalar $\mathcal{C}(u)$ for every $u \in S$ to track states' best-known energy cost during the search. It then proceeds by expanding nodes in order of f-values. Expanding a node x involves extending its partial path to each of its successor states, denoted by $Succ(x)$, thereby generating descendant nodes. During expansion, each descendant node y is checked for feasibility and, if necessary, cost adjustment to ensure the available energy at each state remains within the EV's battery capacity constraints (Artmeier et al. 2010). We can distinguish two special cases: i) if the energy requirement $g(y)$ or estimated total cost $f(y)$ of the descendant node exceeds the available initial energy $\mathcal{E}_{init}$, y is considered infeasible and can be discarded; ii) if the remaining energy at the successor state, computed as $\mathcal{E}_{init} - g(y)$, exceeds the EV's maximum battery capacity (e.g., on sharp downhill segments), then $g(y)$ is adjusted to cap the recuperated energy. This ensures that the remaining energy does not exceed $\mathcal{E}_{max}$, meaning that some recuperated energy cannot be stored due to battery limitations. If a lower-cost path to $s(y)$ is found, the value $\mathcal{C}(s(y))$ is updated, and y is inserted into $Open$ for further exploration. The search terminates when a node with the $goal$ state is extracted from $Open$, with $\mathcal{C}(goal)$ denoting the minimum energy cost. If $\mathcal{C}(goal) = \infty$, it indicates that no feasible path exists under the given initial energy $\mathcal{E}_{init}$.

Pathfinding with unknown $\mathcal{E}_{init}$

A common method to address uncertainty in the initial energy level is to define the energy requirement of an edge (u, v) as a function that maps any possible initial energy $\mathcal{E}_{init} \in [0, \mathcal{E}_{max}]$ at node u to its corresponding energy cost. The energy profile of a path can then be derived by composing the energy profiles of its constituent edges (Baum et al. 2013). As shown in Eisner, Funke, and Storandt (2011) and formally proven in Baum et al. (2020), the energy profiles

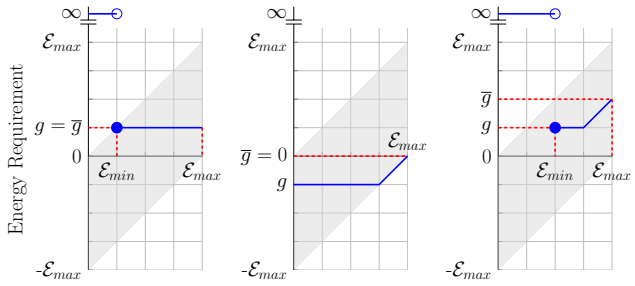

Figure 1: Schematic illustration of energy profiles. Horizontal axis is the $\mathcal{E}_{init}$ range. From left to right: positive-cost link, negative-cost link, and a generic path profile.

of both edges and paths can be represented as piecewise-linear functions with up to two breakpoints and fixed slopes. In our notation, we define the breakpoints of a path profile represented by node x as $(\mathcal{E}_{min}(x), g(x))$ and $(\mathcal{E}_{max}, \overline{g}(x))$, where $\mathcal{E}_{min}(x)$ denotes the minimum energy required to traverse the path, and $g(x)$ and $\overline{g}(x)$ represent the minimum and maximum energy costs, respectively. Here, $\overline{g}(x)$ corresponds to the energy cost when all or part of the recuperated energy cannot be stored due to the capacity limit $\mathcal{E}_{max}$.

Figure 1 illustrates a generic path profile, including examples of negative-cost links, positive-cost links, and generic paths. The shaded area denotes the feasible range. A generic profile consists of two distinct segments:
(i) $\mathcal{E}_{init} < \mathcal{E}_{min}$: the path is not traversable, and the energy cost is considered infinite;
(ii) $\mathcal{E}_{init} \geq \mathcal{E}_{min}$: the path is traversable with a minimum energy cost of $g(x)$ (first segment), but may increase linearly up to $\overline{g}(x)$ as $\mathcal{E}_{init}$ increases (second segment). In the case of negative-energy links, for example, the path can be traversed even with $\mathcal{E}_{init} = 0$. However, if $\mathcal{E}_{init} = \mathcal{E}_{max}$, no energy can be recuperated, resulting in a net energy cost of zero. An important property of these energy profiles is that the slope of the second segment is always one. In other words, at higher initial energy levels, the energy cost increases exactly by the amount of energy that the EV is unable to store (Storandt 2012). Also note that the following inequalities always hold: $\overline{g}(x) \geq g(x)$ and $\mathcal{E}_{min}(x) \geq g(x)$, since the EV must have at least $g(x)$ units of energy to be able to cover the path's minimum cost. In simple cases, such as paths without negative-cost edges (e.g., left plot of Figure 1), we may observe flat profiles with $\mathcal{E}_{min}(x) = g(x) = \overline{g}(x)$.

The profile *linking* procedure involves shifting energy profiles in both the *cost* and $\mathcal{E}_{init}$ domains by the energy cost of the added edge, while respecting the battery capacity constraints (Eisner, Funke, and Storandt 2011; Baum et al. 2013; Schönfelder, Leucker, and Walther 2014). Linking a path profile with an edge profile is a straightforward task and can be performed in constant time as we only need to determine new breakpoints after the profiles are linked, given that the resulting path profile follows the generic piecewise-linear form above. However, the specific calculations vary slightly depending on the search direction, whether a forward or backward search is being performed.

Energy-Optimal Profile A* Search

Existing label-correcting profile search methods employ a specialized procedure, called *merge*, to combine multiple energy profiles associated with the same state. In practice, merging two profiles involves scanning the entire range of initial energy level and computing a new profile that represents the lower envelope of the input profiles, ensuring the minimum possible cost is selected for every value of $\mathcal{E}_{init}$, sometimes by discretizing the SoC range (Eisner, Funke, and Storandt 2011). If the merged profile introduces different breakpoints compared to the existing one, the corresponding state will be re-expanded with the new profile. Although each individual path profile can be described using at most two breakpoints, merged profiles may exhibit up to $|S|$ breakpoints (Baum et al. 2020). Despite this bound, the *merge* operation remains a complex and non-trivial task, requiring careful implementation to ensure efficiency.

We now introduce our novel A*-based approach, which leverages efficient pruning rules to streamline profile search. While the method can be applied in both forward and backward directions, we present it here in the conventional forward search formulation. Inspired by recent advances in multi-objective heuristic search (Hernández et al. 2023; Ahmadi et al. 2025), in particular scenarios with negative weights (Ahmadi et al. 2024a; Ahmadi, Raith, and Jalili 2025), we frame energy-optimal profile search as a multi-objective shortest path problem with negative costs (Stewart and White III 1991), where the minimum required energy ($\mathcal{E}_{min}$) and the maximum energy cost ($\overline{g}$) act as additional objectives alongside the primary objective of minimizing energy cost (g). This formulation enables us to efficiently identify and prune unpromising paths during best-first search, effectively bypassing the need for the costly *merge* operation. Before describing the main steps of the algorithm, we introduce the key concept of *dominance* between search nodes.

Definition. A node y is said to be *dominated* by node x if $\mathcal{E}_{min}(x) \leq \mathcal{E}_{min}(y)$, $g(x) \leq g(y)$, and $\overline{g}(x) \leq \overline{g}(y)$.

Intuitively, this means that the energy profile of node y lies entirely above or equal to that of node x across all relevant initial energy levels, and thus y does not offer a lower energy cost at any point in the $(0, \mathcal{E}_{max}]$ range. Figure 2 illustrates three different scenarios involving energy profiles associated with nodes x, y, and z, where the profile of node z (shown in red) is dominated by that of x. For instance, in the middle plot, node z is dominated by node x since there exists no initial energy level at which z yields a lower energy cost. In contrast, nodes x and y are non-dominated, as neither profile entirely dominates the other: each offers a lower cost over some portion of the initial energy range (vertical axis), and therefore both may contribute to the lower envelope.

A pseudocode of our A*-based profile search is presented in Algorithm 2. It starts with initializing a priority queue $Open$, which traditionally stores unexplored (*Open*) nodes of the search, and a cost upper bound $\overline{f}$. It then sets up for every state $u \in S$ a list $\mathcal{X}(u)$, which is responsible for storing all non-dominated nodes expanded with u. Since the search is done in the forward direction, the algorithm generates an initial (zero cost) node associated with $start$ and inserts it into $Open$. The main search starts at line 7. In every iteration, A* extracts from $Open$ a node with the smallest f-value among all nodes present in the queue. Let this extracted node be x. We have $f(x) = g(x) + h(x)$, where $g(x)$ represents the minimum cost of the concrete path from $start$ to $s(x)$. We now discuss our main pruning rule.

Algorithm 2: Energy Profile (Forward) A* Search

Inputs: A problem instance (G, $cost$, h, $start$, $goal$)
Output: A superset of nodes representing optimal profiles

1 $Open \leftarrow \emptyset$, $\overline{f} \leftarrow \infty$
2 $\mathcal{X}(u) \leftarrow \emptyset \; \forall u \in S$
3 $x \leftarrow$ new node with $s(x) = start$
4 $g(x) \leftarrow 0, \overline{g}(x) \leftarrow 0, \mathcal{E}_{min}(x) \leftarrow 0, parent(x) \leftarrow \varnothing$
5 $f(x) \leftarrow h(start)$
6 Add x to $Open$
7 **while** $Open \neq \emptyset$ **do**
8 Extract from $Open$ a node x with the smallest f-value
9 **if** $f(x) \geq \overline{f}$ **then break**
10 $dominated \leftarrow$ false
11 **for each** $y \in \mathcal{X}(s(x))$ **do**
12 **if** $\mathcal{E}_{min}(y) \leq \mathcal{E}_{min}(x)$ **and** $\overline{g}(y) \leq \overline{g}(x)$ **then**
13 $dominated \leftarrow$ true
14 **break**
15 **if** $dominated =$ true **then continue**
16 add x to $\mathcal{X}(s(x))$
17 **if** $s(x) = goal$ **then**
18 $\overline{f} \leftarrow \min(\overline{f}, \max(\mathcal{E}_{min}(x), \overline{g}(x)))$
19 **continue**
20 **for each** $v \in Succ(s(x))$ **do**
21 $y \leftarrow$ new node with $s(y) = v$
22 $cost_e \leftarrow cost(s(x), v)$
23 $g(y) \leftarrow \max(g(x) + cost_e, -\mathcal{E}_{max})$
24 $f(y) \leftarrow g(y) + h(v)$
25 $\mathcal{E}_{min}(y) \leftarrow \max(\mathcal{E}_{min}(x), g(y))$
26 $\overline{g}(y) \leftarrow \max(0, \overline{g}(x) + cost_e)$
27 $parent(y) \leftarrow x$
28 **if** $\mathcal{E}_{min}(y) > \mathcal{E}_{max}$ **or** $\overline{g}(y) > \mathcal{E}_{max}$ **then**
29 **continue**
30 **if** $\overline{g}(y) + h(v) > \mathcal{E}_{max}$ **then continue**
31 $z \leftarrow$ last node in $\mathcal{X}(v)$
32 **if** $\mathcal{E}_{min}(z) \leq \mathcal{E}_{min}(y)$ **and** $\overline{g}(z) \leq \overline{g}(y)$ **then**
33 **continue**
34 Add y to $Open$
35 **return** $\mathcal{X}(goal)$

Dominance pruning rule: Let x and y be two nodes associated with the same state, and without loss of generality, assume that x is extracted after y has been expanded. This implies $f(y) \leq f(x)$ due to A* processing nodes in non-decreasing order of f-values. Given that the primary cost is already ordered by A* (one dimension reduced), node x is dominated by node y if $\overline{g}(y) \leq \overline{g}(x)$ and $\mathcal{E}_{min}(y) \leq \mathcal{E}_{min}(x)$, meaning that x does not offer a better energy cost than y at any $\mathcal{E}_{init}$ within the range. Now let $\mathcal{X}(s(x))$ denote the set of previously expanded (non-dominated) nodes associated with state $s(x)$. Then, each newly extracted node x can be rigorously checked against nodes in $\mathcal{X}(s(x))$ to

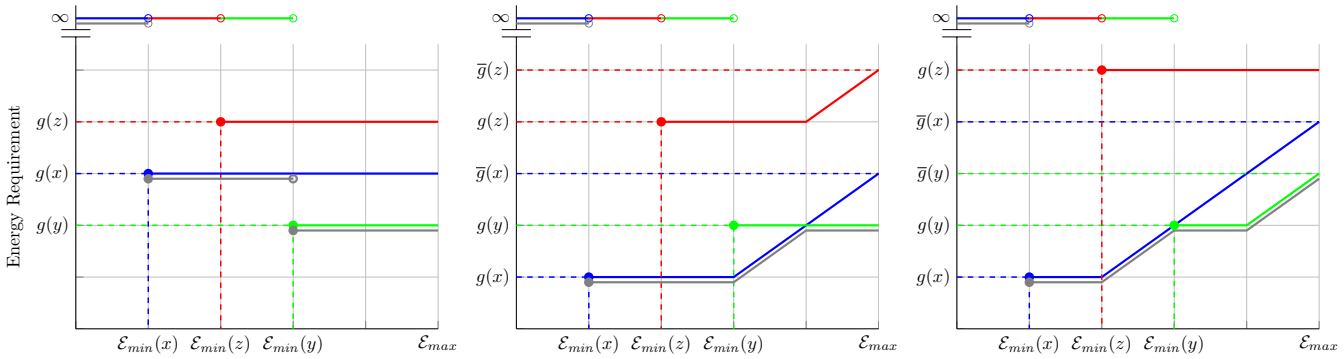

Figure 2: Dominance and non-dominance of energy profiles. In each plot, the profile of z (in red) is dominated by the profile of x (in blue). The profiles of x and y (blue and green) are non-dominated and can both contribute to the lower envelope (in gray).

determine whether x should be pruned (lines 10-15), avoiding expansions that are guaranteed not to contribute to any energy-optimal path. However, there is an important observation in this form of label pruning that distinguishes it from label-correcting approaches: it is possible for a node x to be non-dominated with respect to any individual node in $\mathcal{X}(s(x))$, but still be dominated by the lower envelope formed by the set as a whole. This means that $\mathcal{X}(s(x))$ may contain nodes that do not themselves contribute to the true lower envelope. One might consider a more aggressive pruning strategy by explicitly maintaining the true lower envelope of profiles (via profile merging), as done in label-correcting approaches. However, as we demonstrate in the experimental section, our relaxed pruning rule, though less rigorous, achieves comparable performance to conventional (non-profile) A* search while avoiding the overhead of constructing and maintaining per-state lower envelopes.

Node x is added to the $\mathcal{X}(s(x))$ set if it is deemed non-dominated (line 16). If x is associated with the *goal* state, then a tentative optimal path has been found (lines 17-19), and we can attempt to improve the search upper bound $\overline{f}$ on the solution cost (line 18). Let x be such a tentative solution node. Then, any future node y with estimated cost $f(y)$ no smaller than either $\mathcal{E}_{min}(x)$ or $\overline{g}(x)$ can be safely skipped. This pruning is valid because we would have $f(x) \leq \mathcal{E}_{min}(x) \leq f(y)$ and $f(x) \leq \overline{g}(x) \leq f(y)$ for any such out-of-bounds node y (note that $f(x) = g(x)$ for solution paths). Furthermore, since $f(y) = g(y) + h(s(y)) \leq \overline{g}(y) + h(s(y))$ and likewise $f(y) \leq \mathcal{E}_{min}(y) + h(s(y))$, any solution path obtained by extending y is guaranteed to be dominated by x. Thus, the maximum of $\mathcal{E}_{min}(x)$ and $\overline{g}(x)$ can be treated as an upper bound, which the algorithm can dynamically update when a tighter bound is discovered.

If x is not a solution node, it will be expanded through its successor states (lines 20-34). Let v be one such successor state, i.e., $(s(x), v) \in E$, with its energy requirement denoted by $cost_e$. Each descendant node generated during this expansion corresponds to a new energy profile, obtained by vertically shifting the profile of x by $cost_e$ and adjusting the resulting breakpoints to ensure they remain within the battery constraints. As in energy A* search, we compute for each descendant node y its (minimum) energy cost $g(y)$, bounded by $-\mathcal{E}_{max}$ to avoid recuperation more than the battery capacity, and estimated energy cost $f(y)$. We then compute $\mathcal{E}_{min}(y)$, the minimum initial energy required (at *start*) to traverse the extended path. This is calculated as the maximum of $g(y)$ and the minimum energy required to traverse the prefix path represented by x, i.e., $\mathcal{E}_{min}(x)$. In other words, $\mathcal{E}_{min}(y)$ represents the maximum non-negative cumulative energy cost observed over any prefix of y. This greedy strategy ensures that the minimum required energy does not decrease along the path, even when encountering negative-cost links, since the energy needed to traverse the earlier subpath still applies. Consequently, we always have $g(y) \leq \mathcal{E}_{min}(y)$ in our forward profile search.

To compute $\overline{g}(y)$, we shift $\overline{g}(x)$ by adding $cost_e$, while ensuring that the result remains non-negative to avoid the infeasible scenario of starting a path with a fully charged battery and ending with more than 100% SoC. More importantly, this lower-bounding of $\overline{g}(y)$ enables us to greedily capture the maximum energy cost observed among all subpaths of y ending at v, i.e., the suffix of y with the largest (non-negative) cost. This forms a natural pruning rule: if $\overline{g}(y) > \mathcal{E}_{max}$, then there exists a subpath within y that cannot be traversed even with a fully charged battery, implying that y is infeasible. Note that the remaining energy at v when starting with $\mathcal{E}_{init} = \mathcal{E}_{max}$ is given by $\mathcal{E}_{max} - \overline{g}(y)$.

With the energy profile of the descendant node y computed, we then perform two feasibility checks using battery constraints (lines 28, 30): (i) Node y is considered infeasible if its $\mathcal{E}_{min}$ or $\overline{g}$ exceeds the battery capacity $\mathcal{E}_{max}$, meaning the extended path to state v cannot be traversed even with a fully charged battery; (ii) Node y is also pruned if its estimated total energy cost (i.e., maximum suffix cost of y plus the cost-to-go from v to *goal*) exceeds $\mathcal{E}_{max}$.

Descendant nodes can be added to *Open* if their energy profiles fall within the battery constraints, and if they are not dominated by any previous expansions with state v. However, instead of rigorously verifying dominance during expansion, our approach delays this check and performs it *lazily* when nodes are extracted from the queue. In addition, to avoid overloading the queue with dominated labels, we apply a quick, constant-time dominance check during expansion by comparing the descendant node y with the

most recently expanded node associated with the same successor state v (lines 31-33). This pruning helps reduce unnecessary insertions while preserving correctness. A similar lazy strategy has been used in the context of multi-objective search (Hernández et al. 2023; Ahmadi et al. 2024a).

Finally, the algorithm terminates either by surpassing the upper bound $\overline{f}$ or when $Open$ becomes empty, and returns $\mathcal{X}(goal)$: a superset of nodes associated with energy-optimal paths between $start$ and $goal$. If $\mathcal{X}(goal)$ is empty upon termination, no feasible solution exists for the given problem instance. To choose form optimal paths, one can compute the minimum energy cost for any given initial energy level $\mathcal{E}_{init}$ through the energy profile of nodes in $\mathcal{X}(goal)$. The optimum cost can then be obtained either by selecting a profile that yields the lowest cost at $\mathcal{E}_{init}$, or by computing the lower envelope of all profiles in $\mathcal{X}(goal)$ as a one-time operation. Since all nodes retain full backtracking information, reconstructing solution paths is as simple as following the parent pointers from $goal$ back to $start$. A detailed proof of correctness is provided in Ahmadi and Jalili (2025).

Experimental Analysis

This section evaluates the performance of our energy-optimal profile search framework on real-world road networks. We used 10 large maps from the 9th DIMACS Implementation Challenge[1] as benchmark graphs, with the largest map (Central USA) containing over 14 million states and 34 million edges. The map data includes distance and travel time per edge. We enriched these maps with elevation data from the Shuttle Radar Topography Mission[2]. For each link in the network, we compute energy consumption using the energy model presented in Ahmadi et al. (2024b), which accounts for road gradient, vehicle mass, and speed profiles. We simulate a Nissan Leaf carrying four passengers and equipped with an extended-range 85 kWh battery. All graphs are negative-cycle free. To evaluate performance, we generate 200 random queries per map, yielding a total of 2000 energy-optimal pathfinding queries for our experiments.

Implementation: We implemented four variants of our energy profile search algorithm in C++, alongside standard energy-optimal Dijkstra and A* search algorithms, which work with a known initial SoC. Our four variants include:

- **Pr-A*$_{\text{fw}}$:** A forward energy profile search based on A*, as described in Algorithm 2.
- **Pr-A*$_{\text{bw}}$:** A backward energy profile A* search, which explores the graph from $goal$ to $start$ using predecessors. This variant follows a similar expansion and pruning procedure as the forward version, but differs slightly in how profiles are linked and checked. A pseudocode of this variant is provided in Ahmadi and Jalili (2025).
- **Pr-BA*:** A bidirectional profile search using A*, employing an interleaved strategy that incorporates both forward and backward searches above. Complete paths are constructed by joining forward and backward nodes that meet at the same state. In each iteration, the algorithm selects a node from either direction based on the smallest estimated cost. Nodes can be joined with all explored nodes from the opposite direction (that reach the same state) to form complete paths and potentially update the upper bound if feasible.
- **Pr-BA*$_{\text{par}}$:** A parallel version of Pr-BA*, where each search direction is executed on a separate CPU thread. Each thread performs best-first exploration independently, applying the same pruning and joining rules as in the non-parallel variant.

Pseudocodes for the backward and bidirectional variants are provided in Ahmadi and Jalili (2025). All variants were implemented using the same data structures, and binary heap as the priority queue, along with identical heuristic functions to enable a head-to-head comparison. For the Dijkstra implementation, we equipped it with a potential energy function (identical to the one used in our heuristic function) to enable online reweighting of edge costs without preprocessing, as discussed in Ahmadi et al. (2024b). We assume initial SoC of 100% for non-profile searches. All code was compiled using the `GCC 11.04` compiler with optimization level `-O3`. Experiments were conducted on a machine equipped with an Intel Xeon Platinum 8488C processor @3.2 GHz and 16 GB of RAM. Each instance was run three times, and we report the results corresponding to the run with the median runtime. Our code is publicly available[3].

We report the results for all feasible instances in Table 1, where the instances are grouped into four energy ranges based on their minimum computed energy costs, spanning from 0 to 85 kWh, with at least 240 instances in each group. The results include the minimum, average (arithmetic), and maximum runtimes (in milliseconds) observed for each algorithm across the instances of the group. The reported runtime includes the search time as well as the initialization time of all data structures and priority queue. We also report the average speedup relative to Dijkstra (η), and also the average number of node expansions (last column).

Runtime analysis: From the results, we observe that energy-optimal A* search consistently performs approximately twice as fast as Dijkstra across all instance groups. This highlights the effectiveness of heuristic-guided search, as the slight overhead from heuristic computation is offset by a significant reduction in search time. Comparing the profile-based search methods, it is evident that the forward profile search variant, Pr-A*$_{\text{fw}}$, performs closely to A*. This is likely because their search spaces are largely overlapping, whereas the backward variant operates over a different search space. That said, the backward profile variant, Pr-A*$_{\text{bw}}$, achieves comparable average runtimes to its forward counterpart and is slightly faster in the lower energy range. Compared to A* with known $\mathcal{E}_{init}$, both Pr-A*$_{\text{fw}}$ and Pr-A*$_{\text{bw}}$ are at most 30% slower, demonstrating the efficiency of our proposed technique and its pseudo-polynomial performance on large-scale networks. Surprisingly, both bidirectional variants perform worse than the unidirectional profile search. In particular, Pr-BA* is even slower than Dijkstra. A key reason for this is the bidirectional path match-

[1] http://www.diag.uniroma1.it/challenge9

[2] https://www2.jpl.nasa.gov/srtm/

[3] https://bitbucket.org/s-ahmadi/eospp

Range		Runtime(ms)				#Exp
kWh	Algorithm	Min.	Avg.	Max.	η	$\times 10^3$
0 - 20	Dijkstra	1.9	27.6	152.8	1.00	115
	A*	1.3	15.6	157.5	2.12	42
	Pr-A*$_{\text{fw}}$	2.9	20.6	264.9	1.51	44
	Pr-A*$_{\text{bw}}$	0.8	19.6	262.6	2.03	46
	Pr-BA*	5.3	43.8	516.5	0.74	62
	Pr-BA*$_{\text{par}}$	2.4	30.9	509.0	1.32	62
20 - 40	Dijkstra	10.1	79.5	353.6	1.00	335
	A*	4.1	44.0	184.6	1.98	122
	Pr-A*$_{\text{fw}}$	6.8	55.3	290.8	1.63	125
	Pr-A*$_{\text{bw}}$	3.4	53.1	288.8	1.91	121
	Pr-BA*	14.2	110.9	570.4	0.80	169
	Pr-BA*$_{\text{par}}$	6.0	75.7	543.1	1.34	167
40 - 60	Dijkstra	14.1	160.7	686.8	1.00	645
	A*	5.5	79.8	327.9	2.14	214
	Pr-A*$_{\text{fw}}$	8.1	101.4	513.6	1.77	220
	Pr-A*$_{\text{bw}}$	6.5	101.8	501.3	1.95	222
	Pr-BA*	17.7	202.6	976.7	0.86	304
	Pr-BA*$_{\text{par}}$	10.0	139.1	942.0	1.44	302
60 - 85	Dijkstra	16.3	242.9	1058.2	1.00	992
	A*	10.0	121.7	406.8	2.05	368
	Pr-A*$_{\text{fw}}$	12.3	151.2	549.9	1.69	378
	Pr-A*$_{\text{bw}}$	12.7	152.8	646.6	1.80	382
	Pr-BA*	27.6	298.7	1182.2	0.83	522
	Pr-BA*$_{\text{par}}$	14.4	191.9	1060.1	1.39	516

Table 1: Runtime statistics of the algorithms (in milliseconds) over all feasible instances, separated by four energy ranges. The table also reports speedup (η) relative to Dijkstra, as well as the average number of node expansions.

ing within the fully overlapping search spaces, where the forward and backward frontiers are continuously joined and later evaluated for optimality and potential upper-bound updates, introducing significant computational overhead. The parallel variant, Pr-BA*$_{\text{par}}$, achieves partial performance improvement by enabling concurrent exploration of the forward and backward frontiers. However, it still remains up to 50% slower than the unidirectional variants, highlighting the need for more efficient bidirectional energy profile searches.

Analysis of node expansion: Comparing the number of node expansions, we observe that our forward profile search, Pr-A*$_{\text{fw}}$, expands nearly the same number of nodes as the energy-optimal A* search, while the bidirectional approaches expand approximately 50% more nodes on average. To better understand the distribution of expansions, Figure 3 presents the relative difference in node expansions with respect to energy-optimal A* for the three profile-based variants over all feasible instances. The horizontal axis denotes the optimal energy requirement of the instances (g-values). We observe that the forward variant, Pr-A*$_{\text{fw}}$, incurs less than 10% extra expansions across most instances. In contrast, the backward variant shows far greater variability: depending on the instance, it can require more than 100% additional expansions, or conversely, up to 100% fewer expansions (negative ratios). The bidirectional variant, however,

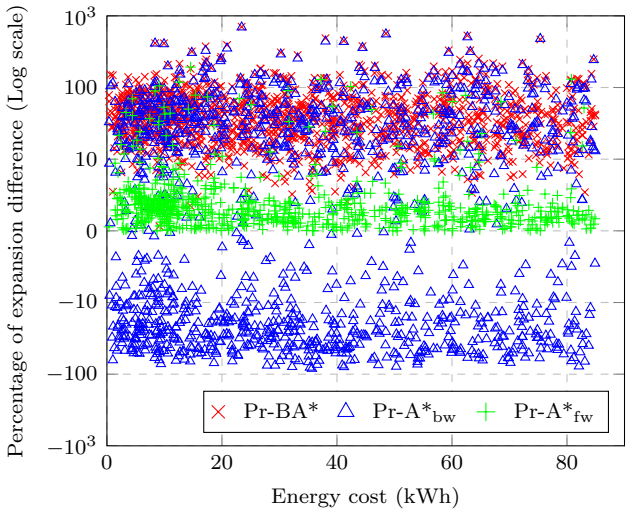

Figure 3: Distribution of expansion differences (%) for profile-search algorithms relative to energy-optimal A*.

consistently exhibits higher expansion overhead, often approaching or exceeding 100% additional node expansions.

Comparison to existing solutions: We first compare our algorithm with the label-correcting approach of Schönfelder, Leucker, and Walther (2014). Their profile A* search is approximately an order of magnitude slower than conventional energy-optimal A*, whereas our approach is less than 30% slower on average. In comparison to the profile search method of Baum et al. (2020), we were unable to reproduce their results or find a publicly available, efficient implementation of their energy profile search method. Therefore, we compare their reported performance against our benchmarks indirectly. Their method is a label-correcting approach based on Dijkstra, which similarly uses a height-induced potential function to obtain reduced energy costs for an EV with 85 kWh battery. According to their reported results, their profile Dijkstra search is up to 50% slower than the standard energy-optimal Dijkstra. Given that our energy profile A* search is up to 70% faster than Dijkstra, we conclude that our approach can be at least twice as fast as the profile search method of Baum et al. (2020), while also being structurally simpler and easier to implement.

Conclusion

This paper presents a novel A*-based approach for energy-optimal profile search for EVs, addressing the practical challenge of energy-efficient pathfinding under unknown initial energy levels. Unlike existing methods that rely on complex label-correcting strategies, our approach leverages multi-objective search and adopts a straightforward label-setting strategy with efficient dominance pruning to propagate energy profiles during the search more effectively. Experimental results show that the proposed approach outperforms existing methods and achieves performance comparable to energy-optimal A* with known initial energy, while offering a simpler yet practical solution for realistic scenarios.

Acknowledgments

This research was supported by the Department of Climate Change, Energy, the Environment and Water under the International Clean Innovation Researcher Networks (ICIRN) program grant number ICIRN000077. Mahdi Jalili is supported by Australian Research Council through projects DP240100963, DP240100830, LP230100439 and IM240100042.

References

Ahmadi, S.; and Jalili, M. 2025. A Fast Heuristic Search Approach for Energy-Optimal Profile Routing for Electric Vehicles. arXiv:2512.01331.

Ahmadi, S.; Raith, A.; and Jalili, M. 2025. A Fast and Simple Algorithm for the Resource Constrained Shortest Path Problem. In Benoit, A.; Kaplan, H.; Wild, S.; and Herman, G., eds., *33rd Annual European Symposium on Algorithms, ESA 2025, September 15-17, 2025, Warsaw, Poland*, volume 351 of *LIPIcs*, 97:1–97:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Ahmadi, S.; Sturtevant, N. R.; Harabor, D.; and Jalili, M. 2024a. Exact Multi-objective Path Finding with Negative Weights. In Bernardini, S.; and Muise, C., eds., *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2024, Banff, Alberta, Canada, June 1-6, 2024*, 11–19. AAAI Press.

Ahmadi, S.; Sturtevant, N. R.; Raith, A.; Harabor, D.; and Jalili, M. 2025. Parallelizing multi-objective A* search. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 35, 131–139.

Ahmadi, S.; Tack, G.; Harabor, D.; and Kilby, P. 2021. Vehicle Dynamics in Pickup-And-Delivery Problems Using Electric Vehicles. In Michel, L. D., ed., *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, volume 210 of *LIPIcs*, 11:1–11:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Ahmadi, S.; Tack, G.; Harabor, D.; Kilby, P.; and Jalili, M. 2024b. Real-Time Energy-Optimal Path Planning for Electric Vehicles. arXiv:2411.12964.

Artmeier, A.; Haselmayr, J.; Leucker, M.; and Sachenbacher, M. 2010. The Shortest Path Problem Revisited: Optimal Routing for Electric Vehicles. In Dillmann, R.; Beyerer, J.; Hanebeck, U. D.; and Schultz, T., eds., *KI 2010: Advances in Artificial Intelligence, 33rd Annual German Conference on AI, Karlsruhe, Germany, September 21-24, 2010. Proceedings*, volume 6359 of *Lecture Notes in Computer Science*, 309–316. Springer.

Baum, M.; Dibbelt, J.; Gemsa, A.; Wagner, D.; and Zündorf, T. 2019. Shortest Feasible Paths with Charging Stops for Battery Electric Vehicles. *Transportation Science*, 53(6): 1627–1655.

Baum, M.; Dibbelt, J.; Pajor, T.; Sauer, J.; Wagner, D.; and Zündorf, T. 2020. Energy-Optimal Routes for Battery Electric Vehicles. *Algorithmica*, 82(5): 1490–1546.

Baum, M.; Dibbelt, J.; Pajor, T.; and Wagner, D. 2013. Energy-optimal routes for electric vehicles. In Knoblock, C. A.; Schneider, M.; Kröger, P.; Krumm, J.; and Widmayer, P., eds., *21st SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2013, Orlando, FL, USA, November 5-8, 2013*, 54–63. ACM.

Bellman, R. 1958. On a routing problem. *Quarterly of applied mathematics*, 16(1): 87–90.

Delling, D.; Goldberg, A. V.; Pajor, T.; and Werneck, R. F. 2017. Customizable Route Planning in Road Networks. *Transp. Sci.*, 51(2): 566–591.

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

Eisner, J.; Funke, S.; and Storandt, S. 2011. Optimal Route Planning for Electric Vehicles in Large Networks. In Burgard, W.; and Roth, D., eds., *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press.

Ford Jr, L. R. 1956. Network flow theory. Technical report, Rand Corp Santa Monica Ca.

Geisberger, R.; Sanders, P.; Schultes, D.; and Delling, D. 2008. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In McGeoch, C. C., ed., *Experimental Algorithms, 7th International Workshop, WEA 2008, Provincetown, MA, USA, May 30-June 1, 2008, Proceedings*, volume 5038 of *Lecture Notes in Computer Science*, 319–333. Springer.

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

Hernández, C.; Yeoh, W.; Baier, J. A.; Felner, A.; Salzman, O.; Zhang, H.; Chan, S.-H.; and Koenig, S. 2023. Multi-objective search via lazy and efficient dominance checks. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 7223–7230.

Johnson, D. B. 1977. Efficient Algorithms for Shortest Paths in Sparse Networks. *J. ACM*, 24(1): 1–13.

Jung, S.; and Pramanik, S. 2002. An Efficient Path Computation Model for Hierarchically Structured Topographical Road Maps. *IEEE Trans. Knowl. Data Eng.*, 14(5): 1029–1046.

Sachenbacher, M.; Leucker, M.; Artmeier, A.; and Haselmayr, J. 2011. Efficient Energy-Optimal Routing for Electric Vehicles. In Burgard, W.; and Roth, D., eds., *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press.

Schönfelder, R.; Leucker, M.; and Walther, S. 2014. Efficient Profile Routing for Electric Vehicles. In Hsu, C. R.; and Wang, S., eds., *Internet of Vehicles - Technologies and Services - First International Conference, IOV, Beijing, China, September 1-3, 2014. Proceedings*, volume 8662 of *Lecture Notes in Computer Science*, 21–30. Springer.

Stewart, B. S.; and White III, C. C. 1991. Multiobjective A*. *J. ACM*, 38(4): 775–814.

Storandt, S. 2012. Quick and energy-efficient routes: computing constrained shortest paths for electric vehicles. In Winter, S.; and Müller-Hannemann, M., eds., *5th ACM SIGSPATIAL International Workshop on Computational Transportation Science 2011, CTS'12, November 6, 2012, Redondo Beach, CA, USA*, 20–25. ACM.