57º SBPO
SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
Gramado • RS
DE 5 A 9 DE OUTUBRO DE 2025

A Comparative Study of RCSPP Algorithms for the Multi-Depot Electric Vehicle Scheduling Problem

Rodrigo Costa da Silva
Universidade Federal do Rio Grande do Sul (UFRGS)
rcosilva@gmail.com

Pablo Cristini Guedes
Universidade Federal do Rio Grande do Sul (UFRGS)
pablo.guedes@ufrgs.br

ABSTRACT

The Multi-Depot Electric Vehicle Scheduling Problem (MDEVSP) is a complex combinatorial optimization problem crucial for sustainable logistics. Column generation is a powerful technique for solving the MDEVSP, where the pricing subproblem often manifests as a Resource Constrained Shortest Path Problem (RCSPP). The efficiency of the overall solution approach heavily depends on the RCSPP algorithm's performance. This paper presents a comparative study of four RCSPP algorithms—Small Label First (SLF), A* (with a null heuristic), State-Space Relaxation (SSR-Inspired), and Advanced Pulse—for solving the MDEVSP subproblem. We evaluate these algorithms on a range of instances, considering both elementary and non-elementary path constraints. The results provide insights into the trade-offs between solution time and path cost, offering guidance for selecting appropriate RCSPP solvers in the context of MDEVSP with electric vehicles.

KEYWORDS. Multi-Depot Electric Vehicle Scheduling; RCSPP; Column Generation; Labeling Algorithms. Paper topics (indicate in order of PRIORITY the paper topic(s))

1. L&T
2. TAG
3. OD

DOI: 10.59254/sbpo-2025-212270
https://proceedings.science/p/212270?lang=pt-br

1. Introduction

The move towards sustainable logistics has increased the use of electric vehicles (EVs). However, integrating EVs into large operations with multiple depots poses significant planning challenges[Liu et al., 2024]. The Multi-Depot Electric Vehicle Scheduling Problem (MDEVSP) encapsulates these complexities, involving the assignment of trips to EVs considering their limited range, recharging requirements, and the coordination across multiple depots and charging facilities [Erdoğan and Miller-Hooks, 2012; Montoya et al., 2017]. Optimizing MDEVSP solutions is vital for minimizing costs and environmental impact, thereby ensuring the viability of EV fleets.

Given its NP-hard nature, Dantzig-Wolfe decomposition with column generation offers a potent exact methodology for the MDEVSP [Desaulniers et al., 2002]. This approach divides the problem into a master problem, which selects optimal vehicle schedules (columns), and pricing subproblems that generate these schedules. For the MDEVSP, the pricing subproblem is a Resource Constrained Shortest Path Problem (RCSPP), tasked with finding feasible, cost-effective EV schedules considering battery constraints, time windows, and recharging logistics [Feillet, 2010].

The overall efficiency of column generation critically depends on the RCSPP solver. The RCSPP, especially its elementary variant (ESPPRC), is NP-hard [Dror, 1994], and various labeling algorithms form the predominant solution paradigm [Irnich and Desaulniers, 2005]. However, their performance varies with RCSPP instance characteristics, and a focused comparison for the MDEVSP context, particularly with EV nuances, warrants further study. This work builds upon prior research in vehicle scheduling, such as the column generation framework for the multiple-depot vehicle type scheduling problem presented by Guedes and Borenstein [2015], extending the investigation to electric vehicles and a detailed comparison of RCSPP subproblem solvers.

This paper evaluates some RCSPP algorithms to determine their effectiveness in solving the MDEVSP subproblem within a column generation framework. We aim to identify the algorithms that offer the best combination of speed and solution quality to enhance the efficiency of MDEVSP solutions. Key contributions include a comparative performance analysis of selected RCSPP algorithms (for both elementary and non-elementary paths) across diverse MDEVSP instances, insights into their strengths and weaknesses based on instance characteristics, and empirically grounded recommendations for algorithm selection.

The paper is organized as follows: Section 2 reviews existing research. Section 3 explains the MDEVSP formulation and the RCSPP solution methods. Section 4 shows the computational experiments and their results. Section 5 discusses what the experiments found. Finally, Section 6 concludes and suggests future research directions.

2. Literature Review

This section reviews literature pertinent to the MDEVSP, column generation applications in EV scheduling, and RCSPP solution algorithms.

2.1. The Multi-Depot Electric Vehicle Scheduling Problem (MDEVSP)

The MDEVSP involves assigning trips to EVs from multiple depots, considering EV-specific constraints like range and recharging [Erdoğan and Miller-Hooks, 2012; Montoya et al., 2017]. Efficient MDEVSP solutions are vital for sustainable logistics. Surveys by Lin et al. [2016] and Perumal et al. [2022] detail models and methods for EV routing/scheduling, highlighting multi-depot and energy replenishment challenges. Prior work by Guedes and Borenstein [2015] addressed the related (non-electric) multiple-depot vehicle type scheduling problem using column generation, providing a foundation for tackling more complex EV variants.

DOI: 10.59254/sbpo-2025-212270
https://proceedings.science/p/212270?lang=pt-br

2.2. Column Generation for MDEVSP and EV Routing

Column generation, based on Dantzig-Wolfe decomposition [Desaulniers et al., 2002], is effective for large-scale VSPs like the MDEVSP. It formulates a master problem selecting vehicle schedules (columns) generated by pricing subproblems. This is widely applied in VRPs [Toth and Vigo, 2014] and adapted for EV routing, addressing constraints like battery management and charging [Schneider et al., 2014; Desaulniers et al., 2016].

2.3. The Resource Constrained Shortest Path Problem (RCSPP) as the Subproblem

The pricing subproblem in MDEVSP column generation is typically an RCSPP or ESPPRC [Dror, 1994; Irnich and Desaulniers, 2005]. It finds a minimum reduced-cost EV schedule respecting resources (time, SoC). Labeling algorithms are standard for RCSPP/ESPPRC [Irnich and Desaulniers, 2005; Righini and Salani, 2008], exploring a state-space (node, resource vector) with dominance rules to prune suboptimal paths. Key classes include: Label-Setting Algorithms (e.g., A* [Hart et al., 1968]; Dijkstra-like when heuristic $h(n) = 0$) anda Label-Correcting Algorithms (e.g., SLF strategies [Ahuja et al., 1993; Desrochers and Soumis, 1988]).

Acceleration techniques involve bounding, heuristic guidance (e.g., Pulse Algorithm [Lozano and Medaglia, 2013]), State-Space Relaxation (SSR) [Irnich et al., 2006; Baldacci et al., 2012], and elementarity enforcement methods [Righini and Salani, 2008]. The algorithms in this study (SLF, A*, SSR-Inspired, Advanced Pulse) are based on these principles.

3. Methodology

This section details the methodological framework: the MDEVSP formulation, the column generation approach, and the RCSPP solution algorithms.

3.1. MDEVSP Formulation

The MDEVSP assigns trips T to a homogeneous EV fleet from depots K to minimize costs. Ω is the set of all feasible single-EV schedules p.

$$\min \quad \sum_{p\in\Omega} c_p x_p \tag{1a}$$

$$\text{s.t.} \quad \sum_{p\in\Omega} a_{tp} x_p = 1, \qquad \forall t \in T \tag{1b}$$

$$\sum_{p\in\Omega_k} x_p \le V_k, \qquad \forall k \in K \tag{1c}$$

$$x_p \in \{0,1\}, \qquad \forall p \in \Omega \tag{1d}$$

where c_p is schedule cost, x_p selects schedule p, a_{tp} indicates if trip t is in p, Ω_k are schedules from depot k, and V_k is vehicle availability at k. EV constraints are handled in the subproblem.

3.2. Column Generation Approach

The MDEVSP is solved via Dantzig-Wolfe decomposition. A Restricted Master Problem (RMP), the LP relaxation of (1) over $\Omega' \subset \Omega$, is solved iteratively. Duals π_t (for (1b)) and μ_k (for (1c)) define pricing subproblems. For each depot k', a subproblem finds $p^* \in \Omega_{k'} \setminus \Omega'$ minimizing reduced cost $\bar{c}_p = c_p^{\text{original}} - \sum_{t\in T_p} \pi_t - \mu_{k'}$. If $\bar{c}_p < 0$, p^* is added to Ω', and the RMP is re-solved [Feillet, 2010].

3.3. The RCSPP Subproblem

The pricing subproblem is an RCSPP on $G = (N, A)$ to find an EV schedule. Arcs have costs and resource consumptions (time, SoC). Key resources are Time and Battery SoC. Solvers for ESPPRC and RCSPP are evaluated, controlled by a 'PermitNonElementary' flag.

3.4. RCSPP Solution Algorithms Implemented

Path length is managed by 'MaxPathLengthAllowed' ($|N|$ for ESPPRC, $|N| \times 5$ for RCSPP).

DOI: 10.59254/sbpo-2025-212270
https://proceedings.science/p/212270?lang=pt-br

3.4.1. Small Label First (SLF) Algorithm

The SLF algorithm, a label-correcting method, processes labels using a deque. This strategy aims to prioritize exploration of promising paths by managing the order of label extensions, often adding labels with better costs to the front of the queue.

Algorithm 1 SLF Algorithm for RCSPP

Initialize $Labels(u) \leftarrow \emptyset, \forall u \in N$; $Q \leftarrow$ deque()
Create L_0 for source s_k; Add L_0 to $Labels(s_k)$ and Q
$iter \leftarrow 0$
while $Q \neq \emptyset$ **and** $iter < MaxIterations_{SLF}$ **do**
    $L_{curr} \leftarrow Q.popleft()$; $u \leftarrow$ node of L_{curr}
    for each outgoing arc (u,v) from u **do**
        $L_{new} \leftarrow$ Extend$(L_{curr},(u,v))$
        if IsFeasible(L_{new}, $PermitNonElementary$, $MaxPathLengthAllowed$) **then**
            if not IsDominated(L_{new}, $Labels(v)$) **then**
                Add L_{new} to $Labels(v)$; Add L_{new} to Q; PruneDominated(L_{new}, $Labels(v), Q$)
    $iter \leftarrow iter + 1$
return Best non-dominated label(s) at destination d_k

3.4.2. A* Algorithm (with $h = 0$)

This A* implementation, with a null heuristic ($h = 0$), functions as Dijkstra's algorithm on the state-space graph. It employs a priority queue to expand labels with the minimum accumulated cost ($g(L)$) first, using a dictionary to store the best-known cost to reach each state.

Algorithm 2 A* Algorithm ($h = 0$) for RCSPP

Initialize $g_costs(state) \leftarrow \infty$; $PQ \leftarrow$ PriorityQueue()
Create L_0 for s_k; $g_costs(state(L_0, PermitNonElementary)) \leftarrow 0$; $PQ.push(L_0)$
$BestSol \leftarrow \emptyset$
while $PQ \neq \emptyset$ **do**
    $L_{curr} \leftarrow PQ.pop()$; $u \leftarrow L_{curr}.node$
    if $g(L_{curr}) > g_costs(state(L_{curr}, PermitNonElementary))$ **then continue**
    if $u = d_k$ **then** Update $BestSol$; **continue**
    for each arc (u,v) from u **do**
        $L_{new} \leftarrow$ Extend$(L_{curr},(u,v))$
        if IsFeasible(L_{new}, $PermitNonElementary$, $MaxPathLengthAllowed$) **then**
            if $g(L_{new}) < g_costs(state(L_{new}, PermitNonElementary))$ **then**
                $g_costs(state(L_{new}, PermitNonElementary)) \leftarrow g(L_{new})$; $PQ.push(L_{new})$
return $BestSol$

The $state(L, PermitNonElementary)$ function adapts state key generation based on whether elementarity is enforced.

3.4.3. SSR-Inspired Algorithm

The SSR-Inspired algorithm utilizes State-Space Relaxation by discretizing resource consumptions into buckets. It maintains a priority queue ordered by actual path costs, but its state-tracking dictionary (g_costs_ssr) uses keys based on the current node and resource *buckets*.

DOI: 10.59254/sbpo-2025-212270
https://proceedings.science/p/212270?lang=pt-br

Algorithm 3 SSR-Inspired Algorithm for RCSPP

Initialize $g_costs_ssr(discrete_state) \leftarrow \infty$; $PQ \leftarrow$ PriorityQueue()
Create L_0 for s_k; $g_costs_ssr(discrete_state(L_0, PermitNonElementary)) \leftarrow 0$; $PQ.push(L_0)$
$BestSol \leftarrow \emptyset$
while $PQ \neq \emptyset$ **do**
    $L_{curr} \leftarrow PQ.pop()$; $u \leftarrow L_{curr}.node$
    if $g(L_{curr}) > g_costs_ssr(discrete_state(L_{curr}, PermitNonElementary))$ **then continue**
    if $u = d_k$ **and** IsResourceFeasibleActual(L_{curr}) **then** Update $BestSol$; **continue**
    for each arc (u,v) from u **do**
        $L_{new} \leftarrow$ Extend($L_{curr}, (u,v)$)
        if IsFeasible(actual resources of L_{new}, $PermitNonElementary$, $MaxPathLengthAllowed$) **then**
            if $g(L_{new}) < g_costs_ssr(discrete_state(L_{new}, PermitNonElementary))$ **then**
                $g_costs_ssr(discrete_state(L_{new}, PermitNonElementary)) \leftarrow g(L_{new})$; $PQ.push(L_{new})$
return $BestSol$

The $discrete_state(L, PermitNonElementary)$ function maps resources to buckets for state identification.

3.4.4. Advanced Pulse Algorithm

This algorithm enhances a basic Pulse framework using a precomputed A*-like heuristic (cost-to-go estimates from a reverse graph Dijkstra if applicable). Pulses are prioritized in a queue based on an f-cost (sum of current cost and heuristic estimate).

Algorithm 4 Advanced Pulse Algorithm for RCSPP

Precompute heuristic $h(u)$ to d_k
Initialize $Labels(u) \leftarrow \emptyset$; $PQ \leftarrow$ PriorityQueue()
Create L_0 for s_k; Add L_0 to $Labels(s_k)$; $PQ.push((g(L_0) + h(s_k), L_0))$
$BestSolCost \leftarrow \infty$
while $PQ \neq \emptyset$ **do**
    $(f_{curr}, L_{curr}) \leftarrow PQ.pop()$; $u \leftarrow L_{curr}.node$
    if $f_{curr} \geq BestSolCost$ **or** IsDominatedAtNode(L_{curr}, $Labels(u)$) **then continue**
    if $u = d_k$ **then**
        if $g(L_{curr}) < BestSolCost$ **then** $BestSolCost \leftarrow g(L_{curr})$
        continue
    for each arc (u,v) from u **do**
        $L_{new} \leftarrow$ Extend($L_{curr}, (u,v)$)
        if IsFeasible(L_{new}, $PermitNonElementary$, $MaxPathLengthAllowed$) **then**
            if $g(L_{new}) + h(v) < BestSolCost$ **and not** IsDominatedAtNode(L_{new}, $Labels(v)$) **then**
                Add L_{new} to $Labels(v)$; PruneDominated(L_{new}, $Labels(v)$)
                $PQ.push((g(L_{new}) + h(v), L_{new}))$
return Best solution corresponding to $BestSolCost$

DOI: 10.59254/sbpo-2025-212270
https://proceedings.science/p/212270?lang=pt-br

3.5. Implementation Details and Experimental Environment

The algorithms were implemented in Python 3, using ‘heapq‘ for priority queues and ‘collections.deque‘ for SLF. Experiments ran on an Intel Core i7 processor with 16GB of DDR3 RAM.

4. Computational Results

This section presents the results of computational experiments conducted to evaluate the four RCSPP algorithms: SLF, A* (with $h = 0$), SSR-Inspired (with 5 buckets per resource), and Advanced Pulse. The evaluation was performed on randomly generated instances, varying in size (number of nodes $N \in \{50, 100, 250, 500, 1000\}$) and density ($D \in \{0.10, 0.50, 0.80\}$). Each algorithm was tested for both elementary (ESPPRC) and non-elementary (RCSPP) path generation.

4.1. Algorithm Characteristics Overview

Table 1 summarizes the core strategies of the implemented solvers.

Table 1: Key Characteristics of the Implemented RCSPP Algorithms.

Characteristic	SLF	A* (h=0)	SSR-Inspired	Advanced Pulse
Algorithm Type	Label-Correcting	Label-Setting	Label-Setting (states)	Hybrid Pulse/A*
Search Strategy	Deque-based	Best-first (g-cost)	Best-first (g-cost)	Best-first (f-cost)
Heuristic	None	$h = 0$	Discretization	Dijkstra (reverse graph)
State Memory	Labels per node	g-costs (node,res[,path])	g-costs (node,buckets[,path])	Labels per node
Path Type Control	Path check & length	State key & length	State key & length	Path check & length

4.2. Results for Elementary Paths (ESPPRC)

The performance of the algorithms when constrained to find elementary paths is presented in Table 2. All reported solutions were validated for resource feasibility.

Table 2: Comparative Results for Elementary RCSPP (ESPPRC) Solvers.

Algorithm	Nodes	Density	Edges	Time (s)	Cost	Path Nodes	Valid
SLF (Elem.)	50	0.10	245	0.0025	−11.20	14	True
A* (Elem., h=0)	50	0.10	245	0.0005	−12.70	10	True
SSR-Inspired Elem. (5b)	50	0.10	245	0.0004	−12.70	10	True
AdvPulse (Elem.)	50	0.10	245	0.0003	−12.70	10	True
SLF (Elem.)	50	0.50	1225	0.0213	−31.80	17	True
A* (Elem., h=0)	50	0.50	1225	0.0029	−46.40	14	True
SSR-Inspired Elem. (5b)	50	0.50	1225	0.0022	−46.40	14	True
AdvPulse (Elem.)	50	0.50	1225	0.0021	−46.40	14	True
SLF (Elem.)	50	0.80	1960	0.0029	−29.40	23	True
A* (Elem., h=0)	50	0.80	1960	0.0683	−58.50	19	True
SSR-Inspired Elem. (5b)	50	0.80	1960	0.0408	−58.50	19	True
AdvPulse (Elem.)	50	0.80	1960	0.0094	−67.10	20	True
SLF (Elem.)	100	0.10	990	0.0037	−32.10	33	True
A* (Elem., h=0)	100	0.10	990	0.0041	−66.00	31	True
SSR-Inspired Elem. (5b)	100	0.10	990	0.0044	−66.00	31	True
AdvPulse (Elem.)	100	0.10	990	0.0054	−66.00	31	True
SLF (Elem.)	100	0.50	4950	0.0529	−58.20	39	True

Continued on next page

DOI: 10.59254/sbpo-2025-212270
https://proceedings.science/p/212270?lang=pt-br

Table 2 continued from previous page

Algorithm	Nodes	Density	Edges	Time (s)	Cost	Path Nodes	Valid
A* (Elem., h=0)	100	0.50	4950	0.0372	−127.90	32	True
SSR-Inspired Elem. (5b)	100	0.50	4950	0.0332	−127.90	32	True
AdvPulse (Elem.)	100	0.50	4950	0.0277	−127.90	32	True
SLF (Elem.)	100	0.80	7920	0.0716	−60.40	39	True
A* (Elem., h=0)	100	0.80	7920	0.0538	−159.20	43	True
SSR-Inspired Elem. (5b)	100	0.80	7920	0.0599	−159.20	43	True
AdvPulse (Elem.)	100	0.80	7920	0.0893	−155.40	40	True
SLF (Elem.)	250	0.10	6225	0.0230	−43.10	75	True
A* (Elem., h=0)	250	0.10	6225	0.0338	−249.60	66	True
SSR-Inspired Elem. (5b)	250	0.10	6225	0.0391	−249.60	66	True
AdvPulse (Elem.)	250	0.10	6225	0.0312	−249.60	66	True
SLF (Elem.)	250	0.50	31 125	0.1048	−66.30	72	True
A* (Elem., h=0)	250	0.50	31 125	0.2791	−327.00	70	True
SSR-Inspired Elem. (5b)	250	0.50	31 125	0.2178	−327.00	70	True
AdvPulse (Elem.)	250	0.50	31 125	0.2507	−308.30	66	True
SLF (Elem.)	250	0.80	49 800	0.4515	−148.10	104	True
A* (Elem., h=0)	250	0.80	49 800	0.5537	−434.10	94	True
SSR-Inspired Elem. (5b)	250	0.80	49 800	0.4125	−434.10	94	True
AdvPulse (Elem.)	250	0.80	49 800	0.8454	−431.50	99	True
SLF (Elem.)	500	0.10	24 950	0.1326	−74.90	139	True
A* (Elem., h=0)	500	0.10	24 950	0.2184	−518.00	125	True
SSR-Inspired Elem. (5b)	500	0.10	24 950	0.2113	−518.00	125	True
AdvPulse (Elem.)	500	0.10	24 950	0.0720	−378.30	88	True
SLF (Elem.)	500	0.50	124 750	1.1241	−186.30	156	True
A* (Elem., h=0)	500	0.50	124 750	0.9937	−673.00	143	True
SSR-Inspired Elem. (5b)	500	0.50	124 750	1.7042	−673.00	143	True
AdvPulse (Elem.)	500	0.50	124 750	1.6472	−683.80	142	True
SLF (Elem.)	500	0.80	199 600	3.4236	−223.20	144	True
A* (Elem., h=0)	500	0.80	199 600	1.3780	−669.90	147	True
SSR-Inspired Elem. (5b)	500	0.80	199 600	1.3141	−669.90	147	True
AdvPulse (Elem.)	500	0.80	199 600	4.7926	−686.50	141	True
SLF (Elem.)	1000	0.10	99 900	2.3742	−142.50	394	True
A* (Elem., h=0)	1000	0.10	99 900	1.5065	−1729.00	374	True
SSR-Inspired Elem. (5b)	1000	0.10	99 900	2.0625	−1729.00	374	True
AdvPulse (Elem.)	1000	0.10	99 900	2.8757	−1801.70	394	True
SLF (Elem.)	1000	0.50	499 500	7.4005	−307.70	278	True
A* (Elem., h=0)	1000	0.50	499 500	128.2035	−1231.40	269	True
SSR-Inspired Elem. (5b)	1000	0.50	499 500	118.6997	−1231.40	269	True
AdvPulse (Elem.)	1000	0.50	499 500	18.4998	−1252.90	258	True
SLF (Elem.)	1000	0.80	799 200	28.7660	−608.10	414	True
A* (Elem., h=0)	1000	0.80	799 200	20.4351	−1829.10	424	True
SSR-Inspired Elem. (5b)	1000	0.80	799 200	19.6391	−1829.10	424	True
AdvPulse (Elem.)	1000	0.80	799 200	86.1225	−1825.10	427	True

DOI: 10.59254/sbpo-2025-212270
https://proceedings.science/p/212270?lang=pt-br

4.3. Results for Non-Elementary Paths (RCSPP)

Table 3 presents the results for the non-elementary RCSPP.

Table 3: Comparative Results for Non-Elementary RCSPP Solvers.

Algorithm	Nodes	Density	Edges	Time (s)	Cost	Path Nodes	Valid
SLF (NaoElem)	50	0.10	245	0.0001	1.40	2	True
A* (NaoElem, h=0)	50	0.10	245	0.0001	1.40	2	True
SSR-Inspired NaoElem (5b)	50	0.10	245	0.0002	1.40	2	True
AdvPulse (NaoElem)	50	0.10	245	0.0001	1.40	2	True
SLF (NaoElem)	50	0.50	1225	0.0020	−17.20	15	True
A* (NaoElem, h=0)	50	0.50	1225	0.0043	−41.20	14	True
SSR-Inspired NaoElem (5b)	50	0.50	1225	0.0036	−41.20	14	True
AdvPulse (NaoElem)	50	0.50	1225	0.0034	−41.20	14	True
SLF (NaoElem)	50	0.80	1960	0.0121	−32.60	20	True
A* (NaoElem, h=0)	50	0.80	1960	0.0134	−53.70	15	True
SSR-Inspired NaoElem (5b)	50	0.80	1960	0.0152	−53.70	15	True
AdvPulse (NaoElem)	50	0.80	1960	0.0215	−53.70	15	True
SLF (NaoElem)	100	0.10	990	0.0079	−29.30	30	True
A* (NaoElem, h=0)	100	0.10	990	0.0090	−81.70	32	True
SSR-Inspired NaoElem (5b)	100	0.10	990	0.0164	−81.70	32	True
AdvPulse (NaoElem)	100	0.10	990	0.0105	−81.70	32	True
SLF (NaoElem)	100	0.50	4950	0.0867	−46.60	32	True
A* (NaoElem, h=0)	100	0.50	4950	0.0608	−127.40	34	True
SSR-Inspired NaoElem (5b)	100	0.50	4950	0.0466	−129.80	35	True
AdvPulse (NaoElem)	100	0.50	4950	0.0755	−127.40	34	True
SLF (NaoElem)	100	0.80	7920	0.0153	−29.30	34	True
A* (NaoElem, h=0)	100	0.80	7920	0.0275	−128.70	35	True
SSR-Inspired NaoElem (5b)	100	0.80	7920	0.0270	−117.50	33	True
AdvPulse (NaoElem)	100	0.80	7920	0.0776	−148.20	35	True
SLF (NaoElem)	250	0.10	6225	0.0740	−32.70	66	True
A* (NaoElem, h=0)	250	0.10	6225	0.0440	−263.90	76	True
SSR-Inspired NaoElem (5b)	250	0.10	6225	0.0363	−260.30	76	True
AdvPulse (NaoElem)	250	0.10	6225	0.0699	−219.20	57	True
SLF (NaoElem)	250	0.50	31 125	0.2190	−127.20	114	True
A* (NaoElem, h=0)	250	0.50	31 125	0.2027	−461.10	101	True
SSR-Inspired NaoElem (5b)	250	0.50	31 125	0.5522	−460.40	100	True
AdvPulse (NaoElem)	250	0.50	31 125	0.9796	−450.70	99	True
SLF (NaoElem)	250	0.80	49 800	2.5682	−150.00	96	True
A* (NaoElem, h=0)	250	0.80	49 800	0.4478	−287.90	65	True
SSR-Inspired NaoElem (5b)	250	0.80	49 800	0.2324	−287.90	65	True
AdvPulse (NaoElem)	250	0.80	49 800	1.2223	−324.60	75	True
SLF (NaoElem)	500	0.10	24 950	0.1540	−67.50	129	True
A* (NaoElem, h=0)	500	0.10	24 950	0.4463	−479.30	114	True
SSR-Inspired NaoElem (5b)	500	0.10	24 950	0.1090	−479.30	114	True

Continued on next page

DOI: 10.59254/sbpo-2025-212270
https://proceedings.science/p/212270?lang=pt-br

Table 3 continued from previous page

Algorithm	Nodes	Density	Edges	Time (s)	Cost	Path Nodes	Valid
AdvPulse (NaoElem)	500	0.10	24950	0.4469	−472.60	116	True
SLF (NaoElem)	500	0.50	124750	1.1562	−187.20	165	True
A* (NaoElem, h=0)	500	0.50	124750	1.0977	−517.30	111	True
SSR-Inspired NaoElem (5b)	500	0.50	124750	0.8254	−517.30	111	True
AdvPulse (NaoElem)	500	0.50	124750	2.7239	−615.80	130	True
SLF (NaoElem)	500	0.80	199600	2.6074	−224.90	142	True
A* (NaoElem, h=0)	500	0.80	199600	7.0336	−2421.70	521	True
SSR-Inspired NaoElem (5b)	500	0.80	199600	4.7789	−2421.70	521	True
AdvPulse (NaoElem)	500	0.80	199600	7.0441	−718.70	152	True
SLF (NaoElem)	1000	0.10	99900	1.1726	−417.10	501	True
A* (NaoElem, h=0)	1000	0.10	99900	2.1095	−2899.60	597	True
SSR-Inspired NaoElem (5b)	1000	0.10	99900	1.3713	−2899.60	597	True
AdvPulse (NaoElem)	1000	0.10	99900	5.0293	−2070.20	436	True
SLF (NaoElem)	1000	0.50	499500	16.3101	−403.20	300	True
A* (NaoElem, h=0)	1000	0.50	499500	4.1136	−1176.30	362	True
SSR-Inspired NaoElem (5b)	1000	0.50	499500	2.7203	−1176.30	362	True
AdvPulse (NaoElem)	1000	0.50	499500	24.1680	−1186.50	285	True
SLF (NaoElem)	1000	0.80	799200	35.9852	−650.80	470	True
A* (NaoElem, h=0)	1000	0.80	799200	72.4728	−3014.00	700	True
SSR-Inspired NaoElem (5b)	1000	0.80	799200	17.5919	−3004.40	706	True
AdvPulse (NaoElem)	1000	0.80	799200	127.2837	−2260.90	536	True

5. Discussion

The computational experiments provided valuable insights into the relative strengths and weaknesses of the evaluated RCSPP algorithms for both elementary and non-elementary pathfinding scenarios, which are critical for their application as pricing subproblem solvers in a column generation framework for the MDEVSP.

5.1. Performance on Elementary Path Problems (ESPPRC)

When strict elementarity was enforced (Table 2), A* (with $h = 0$), SSR-Inspired, and Advanced Pulse consistently outperformed SLF in terms of solution quality (achieved costs). This highlights the effectiveness of their more guided or comprehensive search strategies compared to SLF's simpler deque-based label management, which appears more susceptible to premature convergence to suboptimal paths in the complex ESPPRC state-space.

Regarding computational time for ESPPRC, for instances up to 250 nodes, most algorithms exhibited reasonable speed. Advanced Pulse often showed an edge for smaller instances due to its heuristic guidance. However, as problem size and density increased (e.g., 500-1000 nodes), the overhead associated with Advanced Pulse's heuristic calculation and larger label sets (as it uses a Pulse-like labeling structure) sometimes resulted in longer runtimes compared to A* or SSR-Inspired. A* and SSR-Inspired, while also experiencing significant increases in runtime with scale, often provided a more consistent balance between solution quality and computational effort. The necessity of incorporating path history into the state definition for A* and SSR-Inspired to ensure elementarity is a primary contributor to their computational scaling.

DOI: 10.59254/sbpo-2025-212270
https://proceedings.science/p/212270?lang=pt-br

5.2. Performance on Non-Elementary Path Problems (RCSPP)

Relaxing the elementarity constraint (Table 3) generally allowed algorithms to find paths with substantially lower (better) costs, as expected due to the expanded solution space that includes cyclic paths. This was particularly evident in larger instances where cycles could exploit sequences of arcs with favorable cost and resource consumption.

In this non-elementary setting, A* and SSR-Inspired, with their state definitions modified to exclude path history (relying only on the current node and resource vector/buckets), demonstrated robust performance. SSR-Inspired was particularly noteworthy for its speed on the largest and densest instances, often outperforming A* in time while achieving comparable or very close solution costs. This suggests that its resource discretization effectively prunes the state space without a severe loss in solution quality for non-elementary paths. SLF, while faster than its elementary counterpart, remained generally slower and achieved less competitive costs than A* or SSR-Inspired. Advanced Pulse showed mixed results; while still finding good quality paths, its computational times for large, dense non-elementary instances were often higher, suggesting its precomputed heuristic (designed for elementary paths) offers diminished utility, or its label management becomes a bottleneck in the vastly larger non-elementary search space when combined with a priority queue.

5.3. General Discussion and Implications for MDEVSP

The choice between an elementary (ESPPRC) and non-elementary (RCSPP) subproblem solver in a column generation approach for MDEVSP is a critical design decision. ESPPRCs provide managerially interpretable paths but are computationally more demanding. RCSPPs can be solved faster and potentially lead to better LP relaxations for the master problem, but may require additional mechanisms (e.g., branching rules, post-processing) if strict elementarity is ultimately required for the MDEVSP solution.

Our findings indicate:

- For **ESPPRC**, if solution quality is paramount and time permits, Advanced Pulse can find excellent paths, especially on small to medium instances. For larger instances, A* and SSR-Inspired offer a more reliable balance.
- For **RCSPP (non-elementary)**, SSR-Inspired stands out for its speed, particularly on large, dense problems, while A* (with the adapted state definition) also provides high-quality solutions efficiently.
- The SLF algorithm, in its current implementation, is generally not competitive for either scenario on medium to large instances due to poorer solution quality and/or higher computation times compared to the other methods.

The parameterization of algorithms, such as the number of buckets in SSR-Inspired or the iteration/path length limits for all methods (especially in non-elementary mode), significantly influences performance and termination. These aspects offer avenues for further tuning. Ultimately, the selection of an RCSPP algorithm for an MDEVSP solver should consider the specific characteristics of the MDEVSP instances, the trade-off between subproblem solution time and its impact on the overall column generation convergence, and whether path elementarity is a strict requirement. This study provides empirical data to inform such a selection, highlighting that algorithms like A* and SSR-Inspired, with appropriate state definitions, are strong candidates for both elementary and non-elementary RCSPP subproblems arising in the MDEVSP context.

DOI: 10.59254/sbpo-2025-212270
https://proceedings.science/p/212270?lang=pt-br

6. Conclusions and Future Work

This paper presented a computational comparison of four distinct algorithms—Small Label First (SLF), A* (with $h = 0$), State-Space Relaxation (SSR-Inspired), and Advanced Pulse—for solving the Resource Constrained Shortest Path Problem (RCSPP) and its elementary variant (ESPPRC). These algorithms were evaluated as potential subproblem solvers within a column generation framework for the Multi-Depot Electric Vehicle Scheduling Problem (MDEVSP). Experiments were conducted on a range of instances varying in size and density, considering both elementary and non-elementary path requirements.

Our results indicate that for elementary paths (ESPPRC), A*, SSR-Inspired, and Advanced Pulse generally outperform SLF in solution quality. Advanced Pulse often yields superior costs for smaller instances, while A* and SSR-Inspired provide a more consistent time-cost balance for larger, denser problems. When non-elementary paths (RCSPP) are permitted, A* and SSR-Inspired, benefiting from a state definition that excludes path history, demonstrate strong performance, with SSR-Inspired often being the fastest on large-scale instances while maintaining high solution quality. The choice of algorithm depends significantly on the problem scale, density, and whether elementarity is a strict constraint. SSR-Inspired and A* emerge as robust choices for non-elementary subproblems, while Advanced Pulse, A*, and SSR-Inspired are competitive for elementary subproblems, with trade-offs in computational time for the largest instances.

Several avenues for future research arise from this study. Firstly, the impact of these RCSPP solvers on the convergence rate and overall solution time of the complete MDEVSP column generation and branch-and-price framework warrants a detailed investigation. Secondly, further tuning of algorithmic parameters, such as the number of buckets in SSR-Inspired or more sophisticated iteration/path length limits for non-elementary searches, could yield performance improvements. Exploring more advanced heuristics for the A* and Advanced Pulse algorithms, specifically tailored for EV resource constraints, could also be beneficial. Additionally, extending the comparison to include other state-of-the-art RCSPP algorithms or acceleration techniques from the literature would provide a broader perspective. Finally, investigating the development of hybrid RCSPP approaches that dynamically switch strategies based on instance characteristics or search progress could lead to more robust and efficient subproblem solvers for complex MDEVSP applications.

References

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ. ISBN 0-13-617549-X.

Baldacci, R., Mingozzi, A., and Roberti, R. (2012). Recent exact algorithms for solving vehicle routing problems based on state space relaxation. In *TRISTAN VII - The Seventh Triennial Symposium on Transportation Analysis*, Paris, France.

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M. M., Soumis, F., and Villeneuve, D. (2002). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors, *Column Generation*, p. 35–93. Kluwer Academic Publishers, Boston, MA. ISBN 0-7923-7664-9.

Desaulniers, G., Errico, F., Irnich, S., and Schneider, M. (2016). Exact algorithms for electric vehicle routing problems with time windows. *Operations Research*, 64(6):1388–1405.

Desrochers, M. and Soumis, F. (1988). A reoptimization algorithm for the shortest path problem with time windows. *European Journal of Operational Research*, 35(2):242–254.

DOI: 10.59254/sbpo-2025-212270
https://proceedings.science/p/212270?lang=pt-br

Dror, M. (1994). The shortest path problem with resource constraints and its complexity. *Computers & Operations Research*, 21(2):205–212.

Erdoğan, S. and Miller-Hooks, E. (2012). A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):100–114.

Feillet, D. (2010). A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(4):407–424.

Guedes, M. R. and Borenstein, D. (2015). A column generation based heuristic for the multiple-depot vehicle type scheduling problem. *Gestão & Produção*, 22(2):391–402. ISSN 0104-530X.

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*, SSC-4(2):100–107.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In Crainic, T. G. and Laporte, G., editors, *Column Generation*, volume 4 of *GERAD 25th Anniversary Series*, p. 33–65. Springer US, Boston, MA. ISBN 978-0-387-25485-7.

Irnich, S., Toth, P., and Vigo, D. (2006). The k-cycle constrained s-path problem. *Networks*, 48(2): 85–99.

Lin, J., Zhou, W., and Wolfson, O. (2016). Electric vehicle routing problem. In Shekhar, S., Xiong, H., and Zhou, X., editors, *Encyclopedia of GIS*, p. 553–561. Springer, Cham.

Liu, T., Tan, Z., Luo, M., and Wu, X. H. (2024). Collaboration and resource sharing for the multi-depot electric vehicle routing problem with time windows and dynamic customer demands. *Sustainability*, 16(6). ISSN 2071-1050. URL https://www.mdpi.com/2071-1050/16/6/2700.

Lozano, L. and Medaglia, A. L. (2013). On an exact algorithm for the constrained shortest path problem. *Computers & Operations Research*, 40(1):378–384.

Montoya, A., Guéret, C., Mendoza, J. E., and Villegas, J. G. (2017). The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, 103:87–110.

Perumal, S. G., Nayeem, M. A., BookHary, M. A., Khadem, S. K., and Kharabati, M. (2022). Electric bus planning and scheduling: A state-of-the-art survey. *Renewable and Sustainable Energy Reviews*, 168:112830.

Righini, G. and Salani, M. (2008). New and improved exact algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170.

Schneider, M., Stenger, A., and Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520.

Toth, P. and Vigo, D., editors (2014). *Vehicle Routing: Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition.

DOI: 10.59254/sbpo-2025-212270
https://proceedings.science/p/212270?lang=pt-br