

# A SIMULATED ANNEALING APPROACH TO THE BATTERY ELECTRIC BUS CHARGING PROBLEM

Alexander Brown

College of Engineering  
Utah State University

May 15, 2024

# Outline

Introduction

The Position Allocation Problem Approach With Linear Battery Dynamic

The Simulated Annealing Approach With Linear Battery Dynamics

The Simulated Annealing Approach With Non-Linear Battery Dynamics

## Introduction

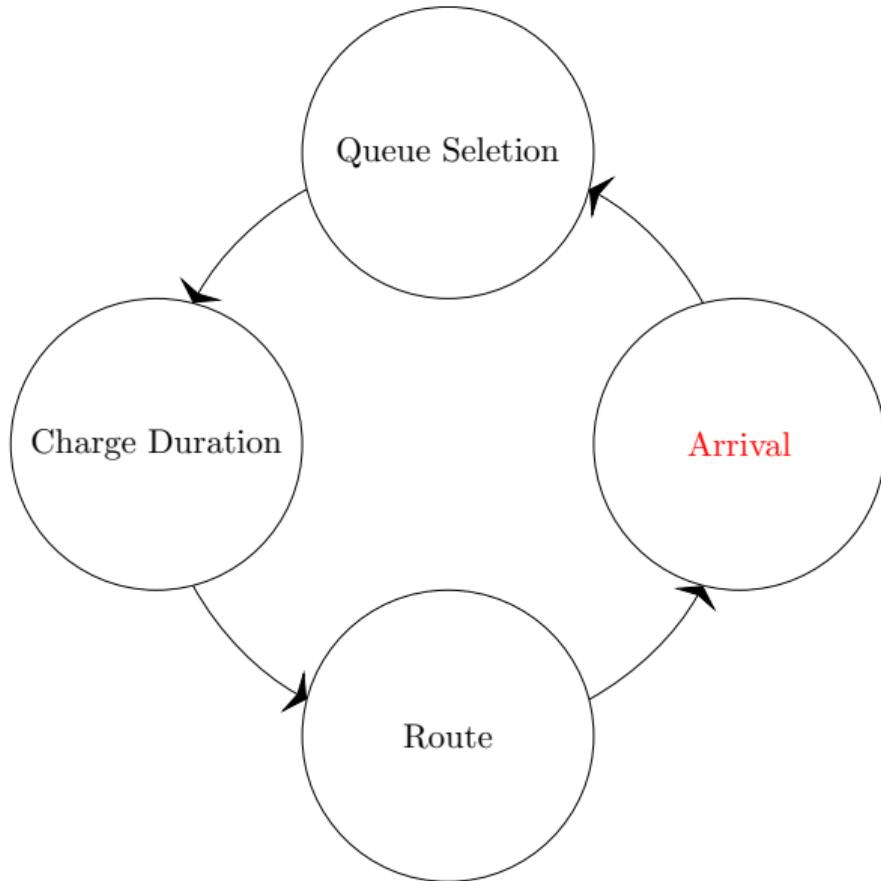
The Position Allocation Problem Approach With Linear Battery Dynamic

The Simulated Annealing Approach With Linear Battery Dynamics

The Simulated Annealing Approach With Non-Linear Battery Dynamics

Ref	Demand	Consumption	Linear Battery	Non-linear Battery	Charger Minimization	Battery Health
[9]	✓					
[7]	✓			✓		
[9]	✓					
[8]	✓					
[1]	✓					
[6]	✓					
[3]	✓	✓	✓	✓	✓	
[2]	✓	✓	✓			
[5]	✓	✓	✓			

## Problem Description



# The Berth Allocation Problem<sup>1</sup>

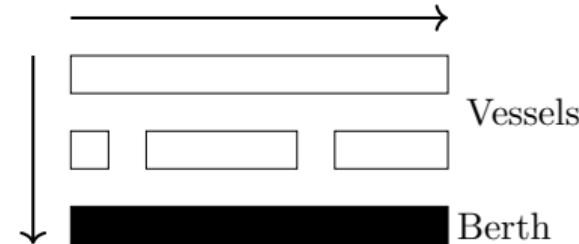


<sup>1</sup><https://www.mdpi.com/2077-1312/11/7/1280>

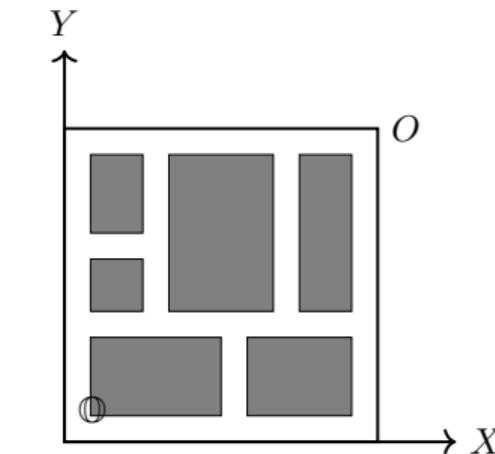
# The Berth Allocation Problem



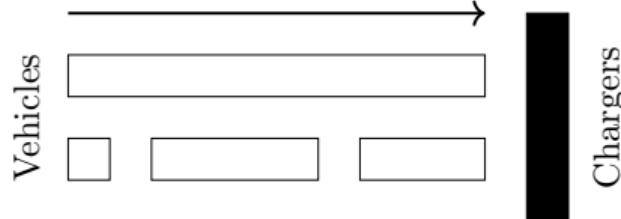
- ▶ Vessels move down toward the quay
  - ▶ Receive service
  - ▶ Exit to the right
- 



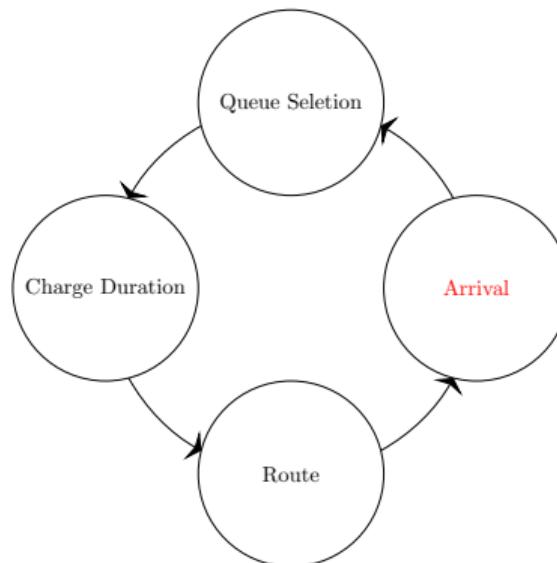
- ▶ A variant of the rectangle packing problem
- ▶ Solves the problem of optimally assigning incoming vessels to be serviced



# The Position Allocation Problem



- ▶ Service flow is left to right
- ▶ Single charger type
- ▶ All arrivals are considered unique
- ▶ Service times are assumed to be known



Introduction

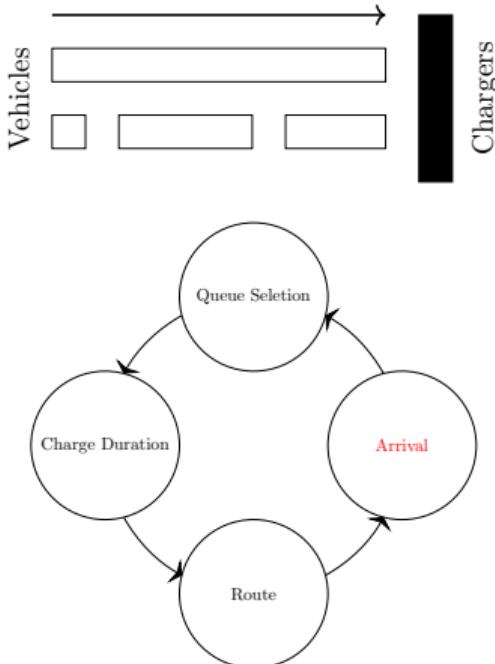
The Position Allocation Problem Approach With Linear Battery Dynamic

The Simulated Annealing Approach With Linear Battery Dynamics

The Simulated Annealing Approach With Non-Linear Battery Dynamics

# Requirements For BEB Implementation

- ▶ Charges must be able to be tracked
- ▶ Service time is unknown
- ▶ Accommodate chargers of different speeds
- ▶ Minimize charger count
- ▶ Minimize consumption cost
- ▶ Encourage slow charger use for battery health



# Constraints



$$\min \sum_{i=1}^{n_V} \sum_{q=1}^{n_Q} \left( w_{iq} m_q + g_{iq} \epsilon_q \right)$$

$$u_j - u_i - s_i - (\sigma_{ij} - 1)T \geq 0$$

$$s_i + u_i = d_i$$

$$v_j - v_i - (\psi_{ij} - 1)n_Q \geq 1$$

$$a_i \leq u_i \leq (T - s_i)$$

$$\sigma_{ij} + \sigma_{ji} \leq 1$$

$$d_i \leq \tau_i$$

$$\psi_{ij} + \psi_{ji} \leq 1$$

$$\eta_i + \sum_{q=1}^{n_Q} g_{iq} r_q - \Delta_i = \eta_{\gamma_i}$$

$$\sigma_{ij} + \sigma_{ji} + \psi_{ij} + \psi_{ji} \geq 1$$

$$\eta_i + \sum_{q=1}^{n_Q} g_{iq} r_q - \Delta_i \geq \nu_{\Gamma_i} \kappa_{\Gamma_i}$$

$$\sum_{q=1}^{n_Q} w_{iq} = 1$$

$$\eta_i + \sum_{q=1}^{n_Q} g_{iq} r_q \leq \kappa_{\Gamma_i}$$

$$v_i = \sum_{q=1}^{n_Q} q w_{iq}$$

# Constraints

$$u_j - u_i - s_i - (\sigma_{ij} - 1)T \geq 0$$

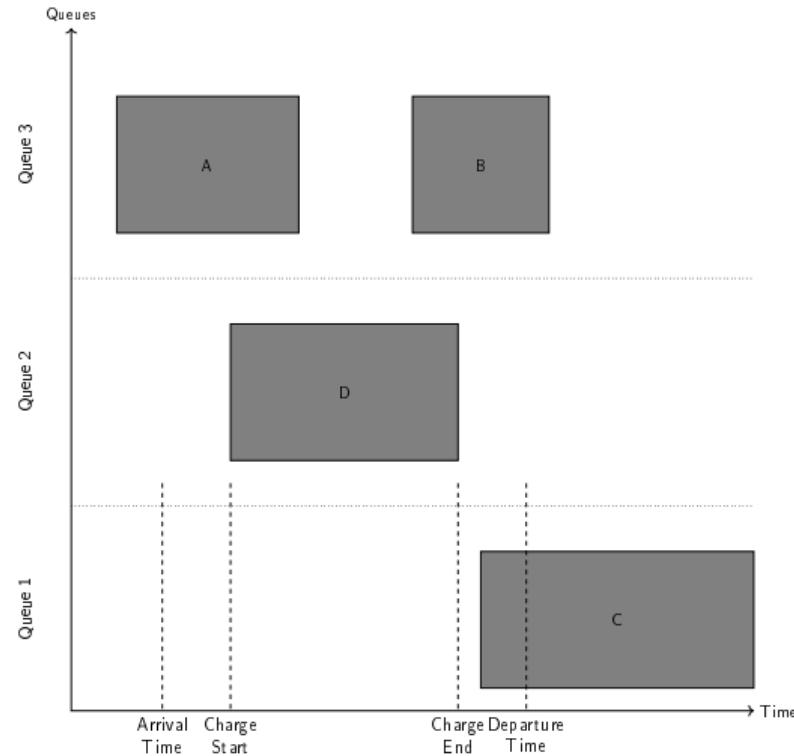
$$v_j - v_i - (\psi_{ij} - 1)n_Q \geq 1$$

$$\sigma_{ij} + \sigma_{ji} \leq 1$$

$$\psi_{ij} + \psi_{ji} \leq 1$$

$$\sigma_{ij} + \sigma_{ji} + \psi_{ij} + \psi_{ji} \geq 1$$

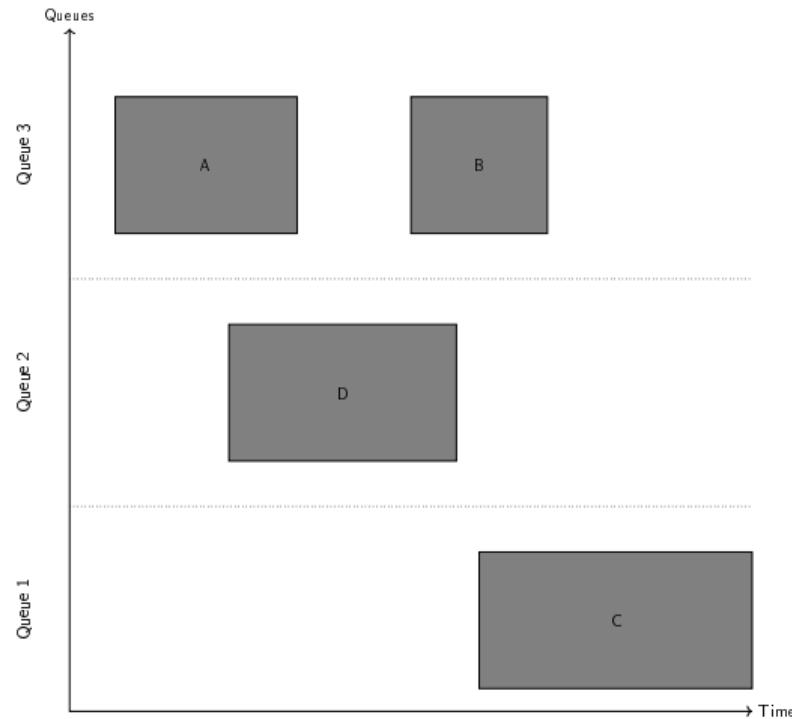
- ▶ Used to ensure that gray rectangles do not overlap
- ▶  $\sigma_{ij}$  establishes temporal ordering when active
- ▶  $\psi_{ij}$  establishes spacial ordering when active



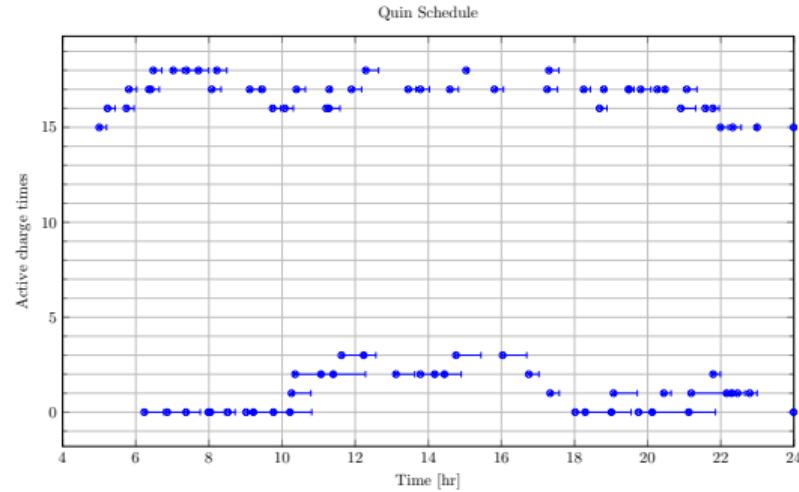
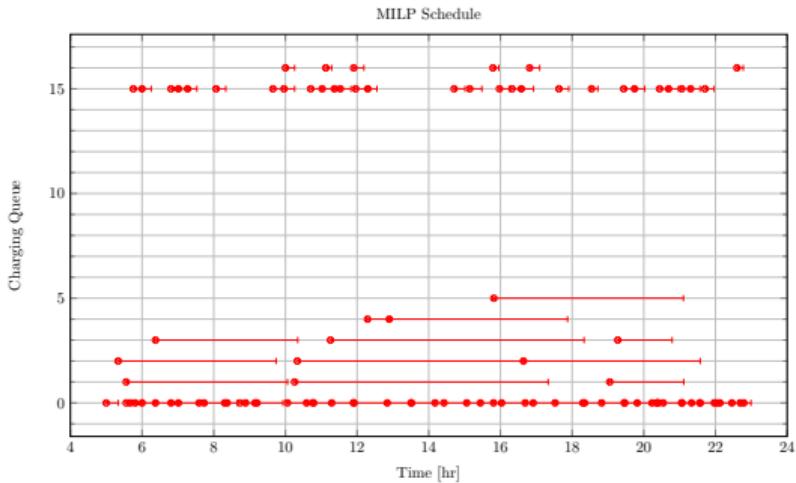
# Parameters



Execution	2 hr
$T$	24 hr
$n_V$	338
$n_A$	35
$\alpha_i$	90%
$\beta_i$	70%
$\nu_i$	20%
$\beta_i$	70%
$m_q$	1000q



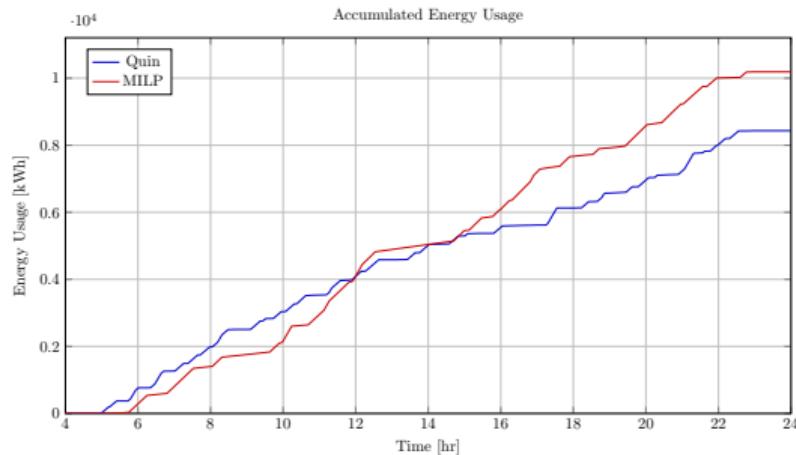
# Schedules



- ▶ Utilized more slow chargers for longer durations
- ▶ Fast chargers utilized more sparingly
- ▶ Extensive use of fast chargers
- ▶ Slow chargers used for short durations

# SOC And Energy Use

	MILP	$Q_{in}$
Mean	265.873	355.93
Min	97.04	0.000
Max	388	368.354



- ▶ SOC of  $Q_{in}$  allowed to drop to 0
- ▶ PAP maintained minimum SOC and final SOC

- ▶ PAP accumulated energy is larger due to minimum SOC constraints

Introduction

The Position Allocation Problem Approach With Linear Battery Dynamic

The Simulated Annealing Approach With Linear Battery Dynamics

The Simulated Annealing Approach With Non-Linear Battery Dynamics

# Simulated Annealing<sup>2</sup>



- ▶ A probabilistic technique for approximating the global optimum of a given function.
- ▶ Often applied to problems that contain many local solutions
- ▶ Three key components:
  - ▶ Cooling Schedule
  - ▶ Acceptance Criteria
  - ▶ Generation Mechanisms

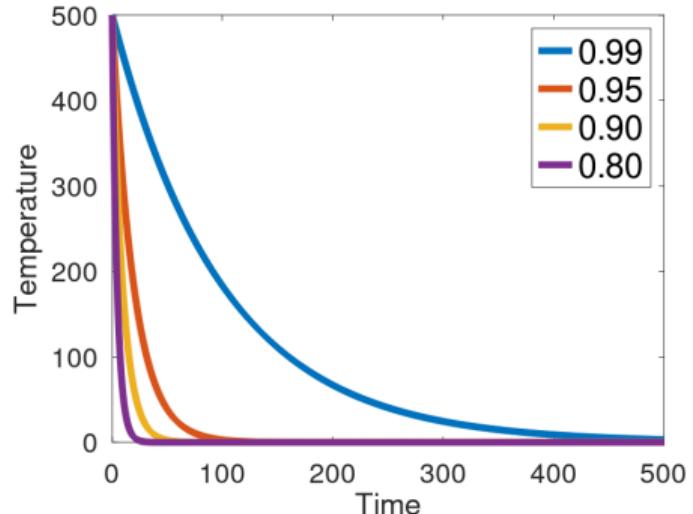
---

<sup>2</sup>[https://en.wikipedia.org/wiki/Simulated\\_annealing](https://en.wikipedia.org/wiki/Simulated_annealing)

# Cooling Schedule



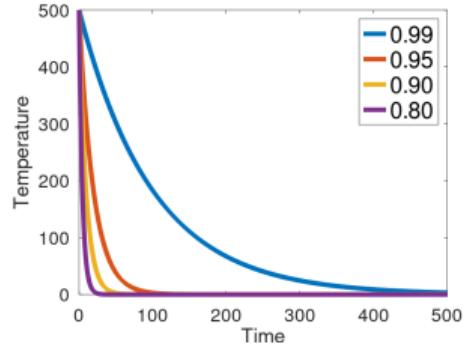
- ▶ The cooling equation models the rate at which the temperature decreases over time in the SA process.
- ▶ The temperature is high, SA encourages exploration. As the temperature decreases, exploitation is encouraged.



$$t_m = \beta t_{m-1}$$

# Acceptance Criteria<sup>3</sup>

$$f(\mathbb{I}, \bar{\mathbb{I}}, t_m) = \begin{cases} 1 & \Delta E > 0 \\ e^{-\frac{\Delta E}{t_m}} & \text{otherwise} \end{cases}$$

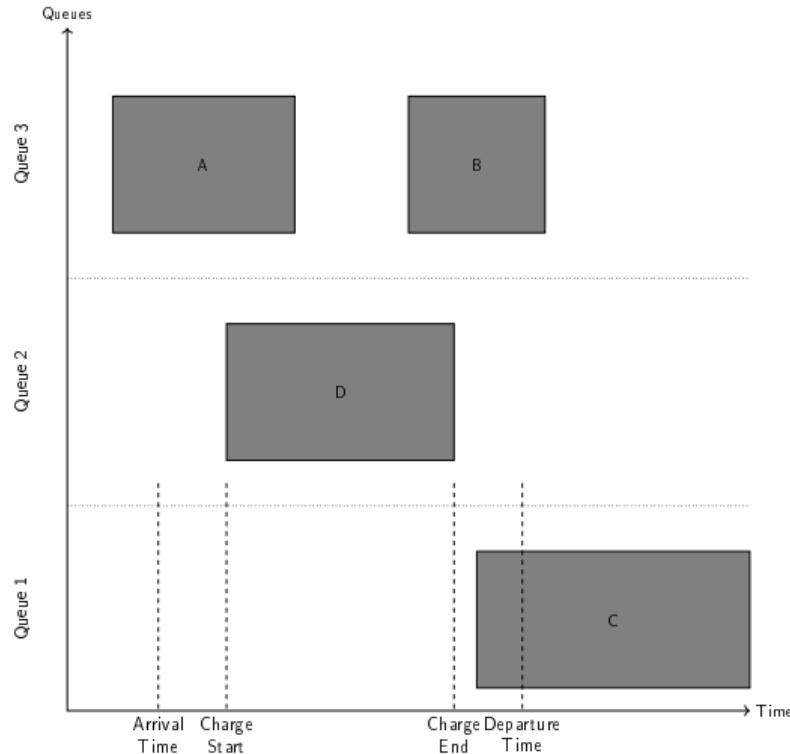


<sup>3</sup>[https://en.wikipedia.org/wiki/Simulated\\_annealing](https://en.wikipedia.org/wiki/Simulated_annealing)

# Generation Mechanisms - Primitive Functions



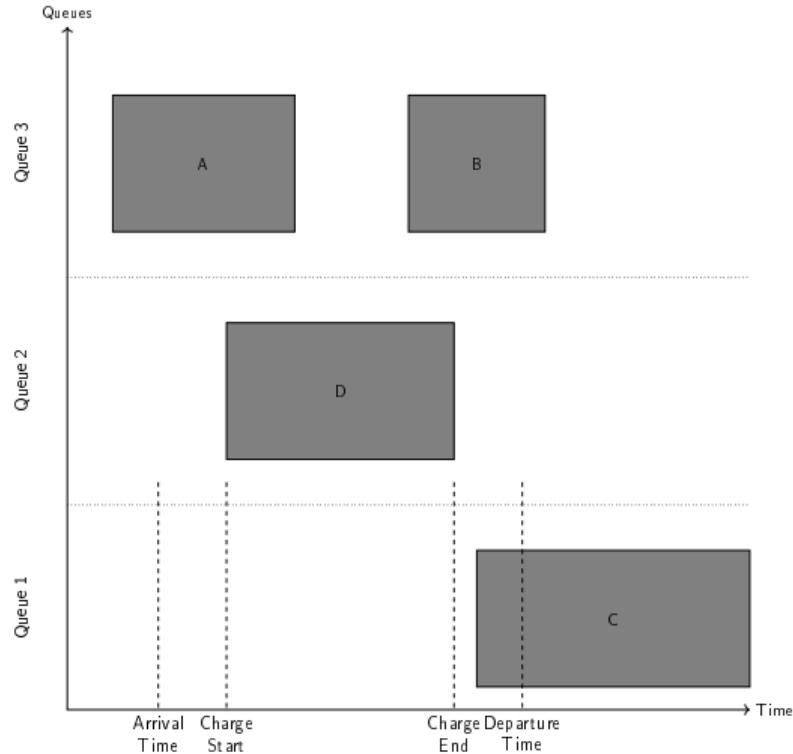
- ▶ New Visit: Move a bus from a wait queue to charge queue
- ▶ Slide Visit: Change the charge duration of a visit
- ▶ New Charger: Move a visit to a new charger
- ▶ Wait: Move a visit to its idle queue
- ▶ New Window: Execute Wait then New Visit primitives



# Generation Mechanisms - Wrapper Functions



- ▶ Charge Schedule Generation:  
Iterate through each visit and execute New Visit
- ▶ Perturb Schedule: Randomly execute one of the primitives with a weighted distribution



# Objective Function



$$J(\mathbb{I}) = z_d p_d + \sum_{i=1}^{n_V} \left[ \epsilon_{q_i} r_{q_i} + z_p \phi_i (\eta_i - \nu_{b_i} \kappa_{b_i}) + z_c r_{q_i} s_i \right]$$

## ► Demand cost

- $p_{T_p, h} = \frac{1}{T_p} \sum_{k=h-\frac{T_p}{dt}+1}^h p_k dt$
- $p_{max} = \max_{k \in [h-\frac{T_p}{dt}+1, h]} p_{T_p, h}$
- $p_d = \max(p_{fix}, p_{max})$

- $\epsilon_{q_i} r_{q_i}$ : Assignment Cost
- $z_p \phi_i (\eta_i - \nu_{b_i} \kappa_{b_i})$ : Penalty Function
- $z_c r_{q_i} s_i$ : Consumption Cost

# Constraints

$$u_j - d_i - (\sigma_{ij} - 1)T \geq 0$$

$$q_j - q_i - 1 - (\psi_{ij} - 1)Q \geq 0$$

$$\sigma_{ij} + \sigma_{ji} \leq 1$$

$$\psi_{ij} + \psi_{ji} \leq 1$$

$$\sigma_{ij} + \sigma_{ji} + \psi_{ij} + \psi_{ji} \geq 1$$

$$s_i = d_i - u_i$$

$$\eta_{\xi_i} = \eta_i + r_{q_i} s_i - \Delta_i$$

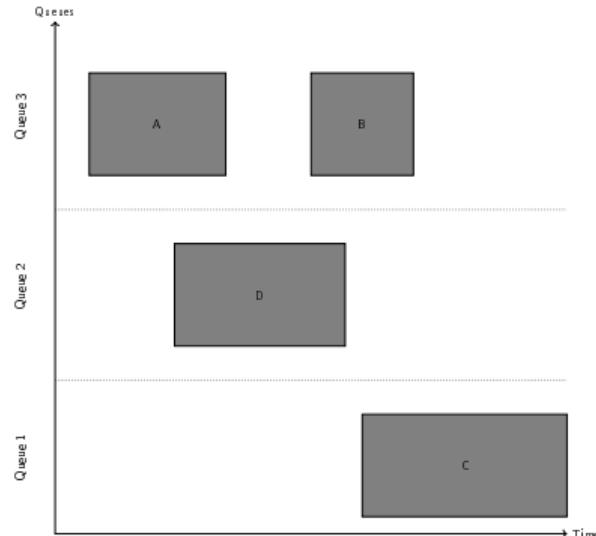
$$\kappa_{\Xi_i} \geq \eta_i + r_{q_i} s_i$$

$$a_i \leq u_i \leq d_i \leq e_i \leq T$$

# Parameters

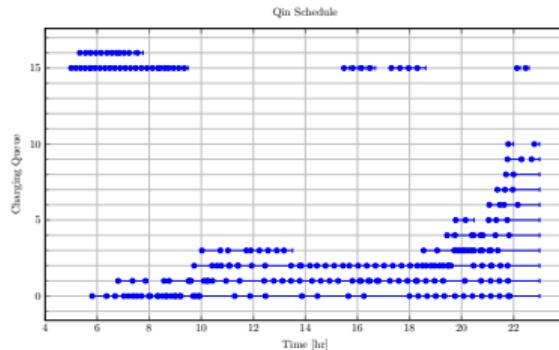
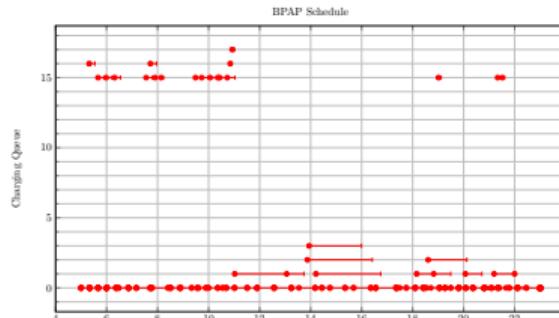
Model	Execution Time [s]
MILP	1900
Quick	1532.8
Heuristic	1916

- ▶  $T_0 = 90000$
- ▶  $|t| = 3797$
- ▶  $\beta = 0.997$
- ▶  $n_K = 500$

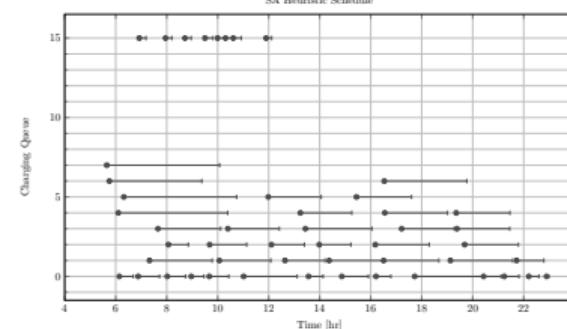
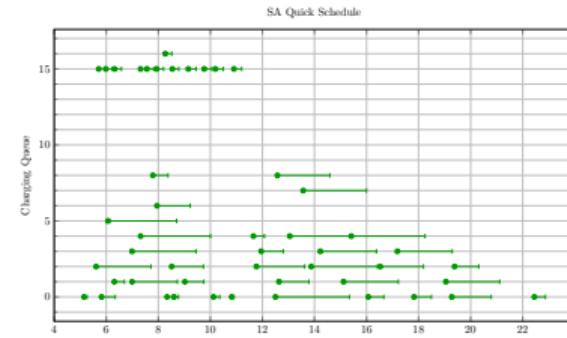


# Schedule

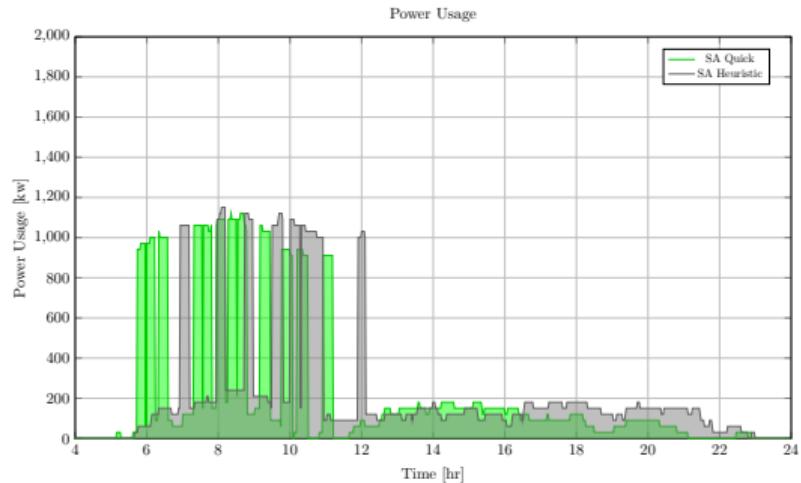
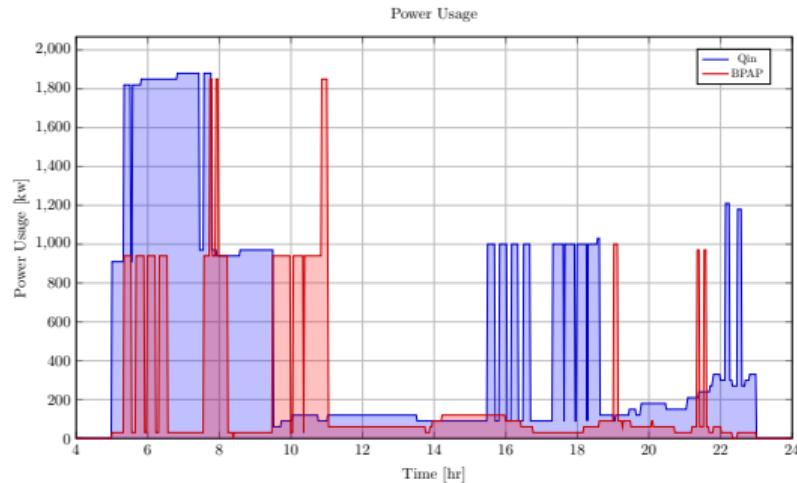
- ▶ Qin heavily utilizes fast charger
- ▶ PAP emphasizes slow queues, but utilizes fast chargers readily



- ▶ SA techniques heavily utilize slow charging
- ▶ SA techniques utilize fast chargers more sparingly



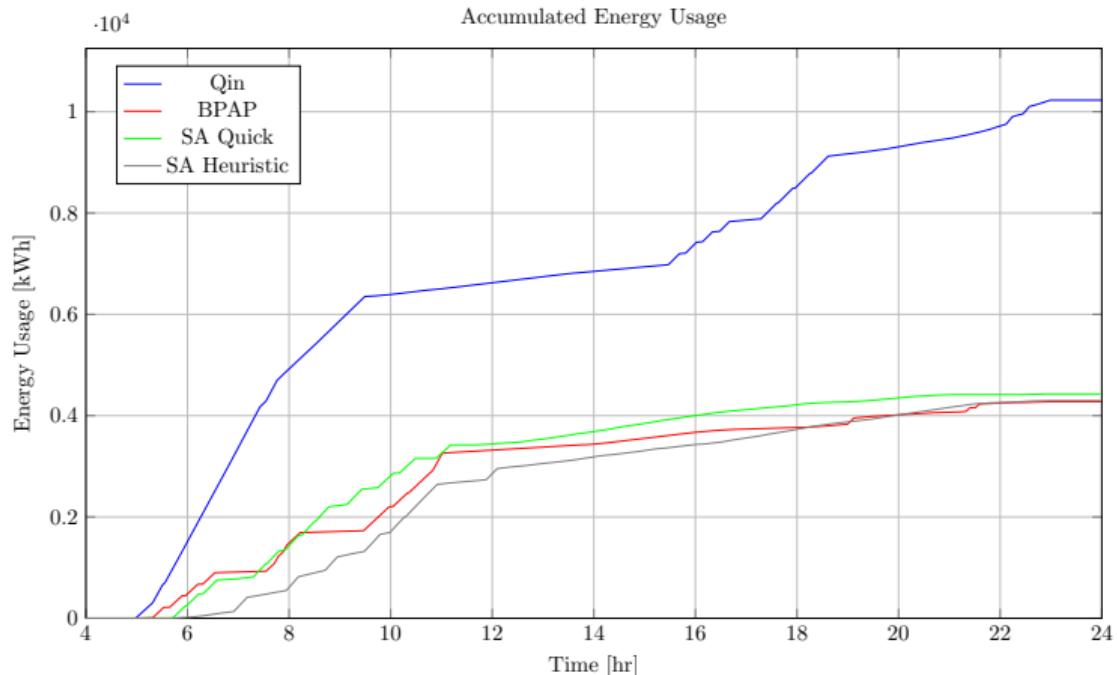
# Power



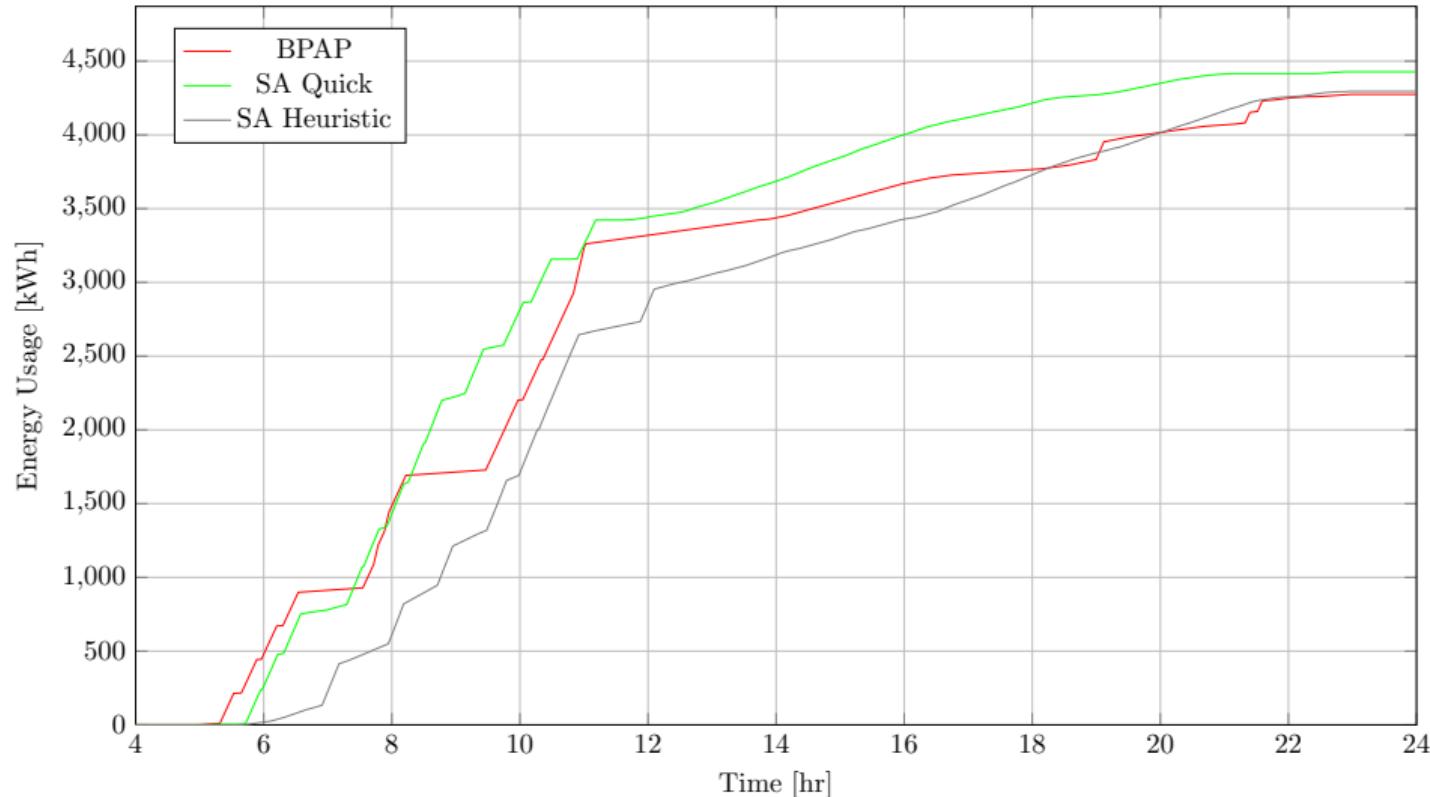
- ▶ Heuristic SA maintained the lowest demand peak
- ▶ Quick SA peak demand comparable to the PAP peak

# Energy and SOC

- ▶ PAP maintained lowest energy consumption
- ▶ Second lowest consumed energy by the heuristic SA (185 kW difference)
- ▶ Trade off of the lower peak demand



Accumulated Energy Usage



Introduction

The Position Allocation Problem Approach With Linear Battery Dynamic

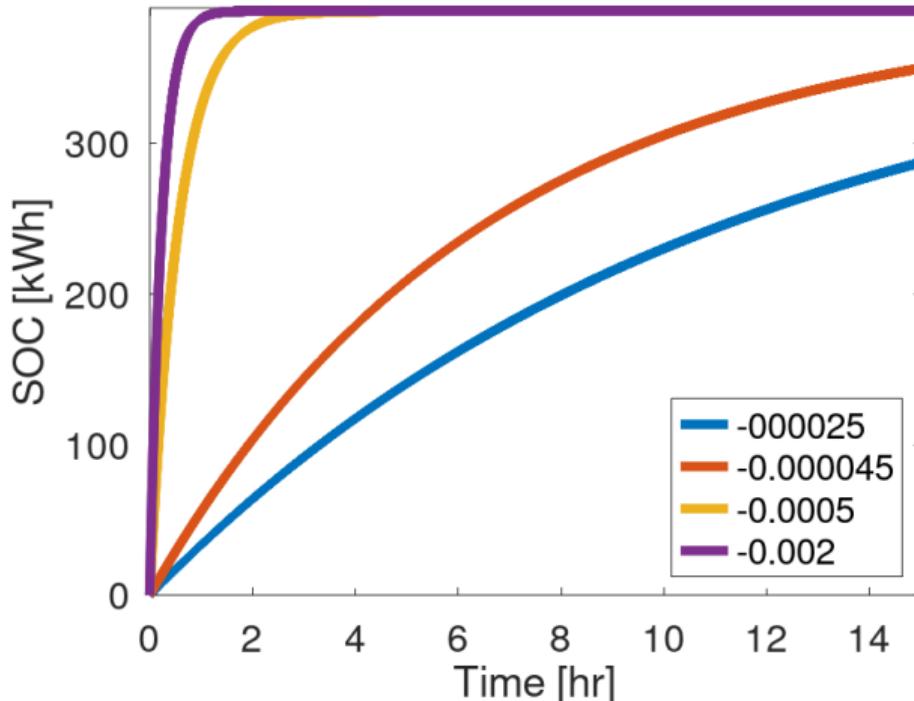
The Simulated Annealing Approach With Linear Battery Dynamics

The Simulated Annealing Approach With Non-Linear Battery Dynamics

# Introduction



- ▶ Higher fidelity in approximating charge at high SOC
- ▶ Implemented in SA for simplicity

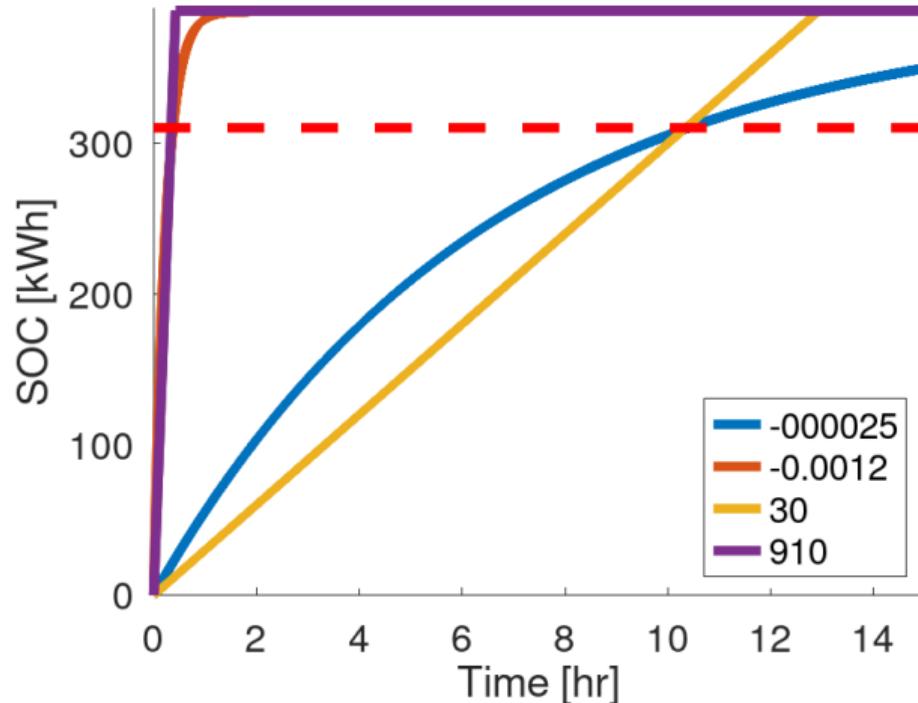


# Non-Linear Battery Dynamics Model



$$\eta_{\xi_i} = \bar{a}_q \eta_i - \bar{b}_q \kappa_{\Xi_i}$$

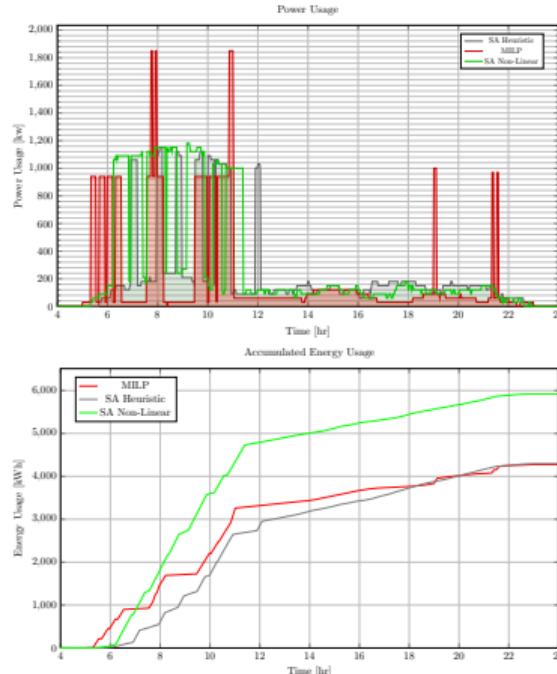
$$\bar{a}_q = e^{a_q dt} \quad \bar{b}_q = e^{a_q dt} - 1$$



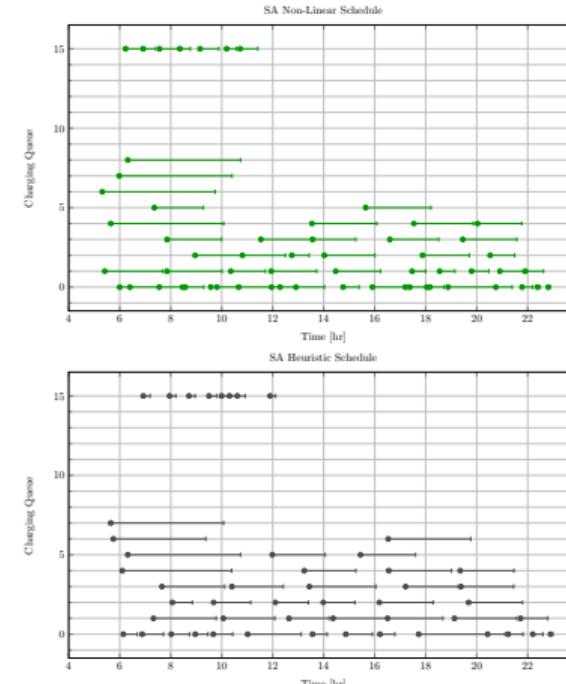
# Results



- ▶ Minimized demand power
- ▶ Larger energy consumption due to fast charger duration



- ▶ Similar schedule output, longer fast charger durations due to nonlinear model
- ▶ Minimum SOC of 85 kWh



# Appendix

# Mixed Integer Linear Programming

$$\max J = \sum_j c_j x_j + \sum_k d_k y_k$$

$$\text{subject to } \sum_j a_{ij} x_j + \sum_k g_{ik} y_k \leq b_i \quad (i = 1, 2, \dots, m)$$

$$x_j \geq 0 \quad (j = 1, 2, \dots, n)$$

$$y_k \in \mathbb{Z}^+ \quad (k = 1, 2, \dots, n)$$

- ▶  $J$ : Objective function
- ▶  $x_j \in \mathbb{R}$  and  $y_k \in \mathbb{Z}^+$ : Decision Variables
- ▶  $c_j, d_k, a_{ij}, g_{ik}, b_i \in \mathbb{R}$ : Parameters

# SA Primitive Generators



College of Engineering  
UtahStateUniversity

---

**Algorithm 0:** New visit algorithm

---

**Algorithm:** New Visit**Input:**  $\mathbb{S}$ **Output:**  $\bar{\mathbb{S}}$ **begin**

```
1       $i \leftarrow \mathbb{S}_i;$                                 /* Extract visit index */
2       $a \leftarrow \mathbb{I}_{i.a};$                             /* Extract the arrival time for visit  $i$  */
3       $e \leftarrow \mathbb{I}_{i.e};$                             /* Extract the departure time for visit  $i$  */
4       $q \leftarrow \mathbb{I}_{i.q};$                             /* Extract the current charge queue for visit  $i$  */
5       $\bar{q} \leftarrow \mathcal{U}_Q;$                          /* Select a random charging queue with a uniform distribution */
6       $C \leftarrow \mathcal{U}_{\mathbb{C}_q};$                       /* Select a random time slice from  $\mathbb{C}_q$  */
7      if  $(\bar{C}, \bar{u}, \bar{d}) \leftarrow \text{findFreeTime}(C, i, q, a, e) \notin \emptyset$  then    /* If there is time available in  $C_q^j$  */
8          |   return  $(i, (\bar{q}, \bar{u}, \bar{d}), \bar{C})$                                 /* Return visit */
9      end
10     return  $(\emptyset);$                                 /* Return nothing */
11
12 end
```

---

---

**Algorithm 1:** Slide Visit Algorithm

---

**Algorithm:** Slide Visit**Input:**  $\mathbb{S}$ **Output:**  $\bar{\mathbb{S}}$ 

```
1 begin
2    $(i, \mathbb{I}, \bar{C}) \leftarrow \text{Purge}(\mathbb{S});$            /* Purge visit  $i$  from charger availability matrix */
3    $C \leftarrow \bar{C}_{i,q_i};$                          /* Get the time availability of the purged visit */
4   /* If there is time available in  $C$ 
5   if  $(\bar{C}, \bar{u}, \bar{d}) \leftarrow \text{findFreeTime}(C, \mathbb{S}_i, \mathbb{I}_q, \mathbb{I}_{i.a}, \mathbb{I}_{i.e}) \notin \emptyset$  then
6     | return  $(i, \mathbb{I}, (\mathbb{I}_{i.q_i}, \bar{u}, \bar{d}), \bar{C})$           /* Return updated visit */
7   end
8   return  $(\emptyset);$                                 /* Return nothing */
9 end
```

---

---

## Algorithm 2: New Charger Algorithm

---

**Algorithm:** New Charger

**Input:**  $\mathbb{S}$

**Output:**  $\bar{\mathbb{S}}$

1 **begin**

```
2    $(i, \mathbb{I}, \bar{\mathbb{C}}) \leftarrow \text{Purge}(\mathbb{S});$            /* Purge visit  $i$  from charger availability matrix */
3    $q \leftarrow \mathcal{U}_Q;$            /* Select a random charging queue with a uniform distribution */
4   if  $(\bar{\mathbb{C}}, \bar{u}, \bar{d}) \leftarrow \text{findFreeTime}(\bar{\mathbb{C}}_{i,q}, \mathbb{S}_i, \mathbb{I}_q, \mathbb{I}_{i,u}, \mathbb{I}_{i,d}) \notin \emptyset$  then /* If there is time available in
       $C_q$  */
      /* Return visit, note  $u$  and  $d$  are the original initial/final charge times. */
      return  $(i, \mathbb{I}, (q, \mathbb{I}_{i,u}, \mathbb{I}_{i,d}), \bar{\mathbb{C}})$ 
6   end
7   return  $(\emptyset);$            /* Return nothing */
8 end
```

---

---

**Algorithm 3:** Wait algorithm

---

**Algorithm:** Wait**Input:**  $\mathbb{S}$ **Output:**  $\bar{\mathbb{S}}$ 

```
1 begin
2    $(i, \mathbb{I}, \bar{\mathbb{C}}) \leftarrow \text{Purge}(\mathbb{S});$            /* Purge visit  $i$  from charger availability matrix */
3    $\bar{\mathbb{C}}'_{\mathbb{I}_{i,r_i}} \leftarrow \mathbb{C}' \cup \{[\mathbb{I}_{i.b}, \mathbb{I}_{i.e}]\};$  /* Update the charger availability matrix for wait queue  $\bar{\mathbb{C}}_{i,q_i}$  */
4   return  $(i, \mathbb{I}, (\mathbb{I}_{i.b}, \mathbb{I}_{i.a}, \mathbb{I}_{i.e}), \bar{\mathbb{C}})$            /* Return visit */
5 end
```

---

---

## Algorithm 4: New window algorithm

---

**Algorithm:** New Window

**Input:**  $\mathbb{S}$

**Output:**  $\bar{\mathbb{S}}$

```
1 begin
2   |  $\bar{\mathbb{S}} \leftarrow \text{Wait}(\mathbb{S})$ ;           /* Assign visit to its respective idle queue */
3   | if  $\bar{\mathbb{S}} \leftarrow \text{NewVisit}(\bar{\mathbb{S}}) \neq \emptyset$  then
4   |   | return  $\bar{\mathbb{S}}$                          /* Add visit  $i$  back in randomly */
5   | end
6   | return ( $\emptyset$ );                          /* Return nothing */
7 end
```

---

# SA Wrapper Functions



College of Engineering  
UtahStateUniversity

---

## Algorithm 5: Charge schedule generation algorithm

---

**Algorithm:** Candidate Solution Generator

**Input:**  $\mathbb{S}$

**Output:**  $\bar{\mathbb{S}}$

```
1 begin
2   /* Select an unscheduled BEB visit from a randomly indexed set of visits */ 
3   foreach  $\mathbb{I}_i \in \mathbb{I}$  do
4     |  $(i, \bar{\mathbb{I}}, \bar{\mathbb{C}}) \leftarrow \text{NewVisit}((\mathbb{I}_i, \mathbb{I}, \mathbb{C}))$ ;           /* Assign the bus to a charger */
5   end
6   return  $(0, \bar{\mathbb{I}}, \bar{\mathbb{C}})$ 
7 end
```

---

# Charge Schedule Perturbation



---

**Algorithm 6:** Perturb schedule algorithm

---

**Algorithm:** Perturb Schedule

**Input:**  $\mathbb{S}$   
**Output:**  $\tilde{\mathbb{S}}$

```
1 begin
2      $p \leftarrow [\text{false}; n_A]$ ;                                /* Create vector to track priority routes */
3      $y^i \leftarrow [1.0; n_V]$ ;                                /* Create weight vector for index selection */
4     /* Loop through the visits in reverse order */                */
5     foreach  $\mathbb{I}_i \leftarrow \mathbb{I}_{|\mathbb{I}|} \text{ TO } \mathbb{I}_1$  do
6         /* If the current visit is part of a priority route */      */
7         if  $p_{\mathbb{I}_i.b} = \text{true}$  then
8              $y_{\mathbb{I}_i}^i = y_{\mathbb{I}_i,\xi}^i$ ;                            /* */
9         end
10        /* Else if the current visit's SOC does below the allowed threshold */ */
11        else if  $\mathbb{I}_{i.\eta} \leq \nu_{\mathbb{I}_i.b} \kappa_{\mathbb{I}_i.b}$  then
12             $p_{\mathbb{I}_i.b} = \text{true}$ ;                                /* Indicate the current BEB's routes are to be prioritized */
13             $y_{\mathbb{I}_i}^i = \kappa_{\mathbb{I}_i.b} (\nu_{\mathbb{I}_i.b} \kappa_{\mathbb{I}_i.b} - \mathbb{I}_{i.\eta})$ ; /* Calculate the weight of the current visit */
14        end
15    end
16     $\mathbb{I}_i \leftarrow \mathcal{W}_{\mathbb{I}}^{y^i}$ ;                                /* Select an index with a weighted distribution */
17     $i \leftarrow \mathbb{I}_i$ ;                                    /* Extract visit index */
18     $y^P \leftarrow [y_1^P, y_2^P, \dots]$ ;                  /* Define the weight of each primitive generator */
19     $PGF \leftarrow \mathcal{W}_{[1, n_G]}^{y^P}$ ;                /* Select a generator function with weighted distribution */
20     $\tilde{\mathbb{S}} \leftarrow PGF((i, \mathbb{I}, \mathbb{C}))$ ;          /* Execute the generator function */
21    return ( $0, \mathbb{I}, \tilde{\mathbb{C}}$ )
22 end
```

---

# SA Thesis Results



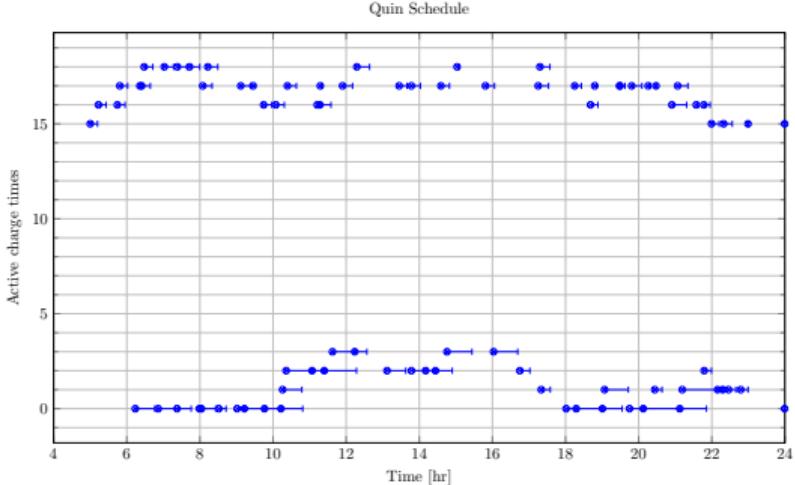
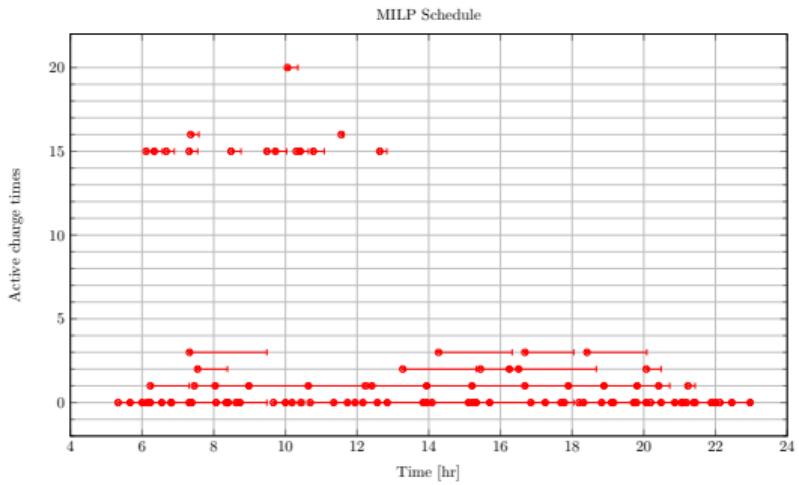
College of Engineering  
UtahStateUniversity

## Parameters

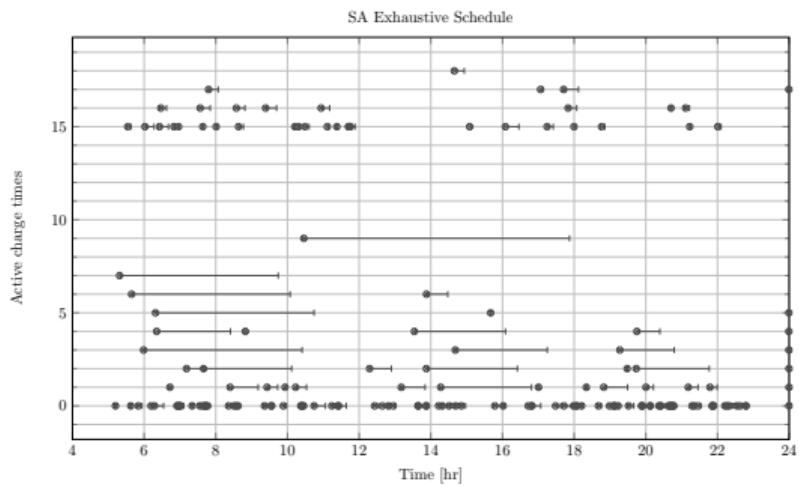
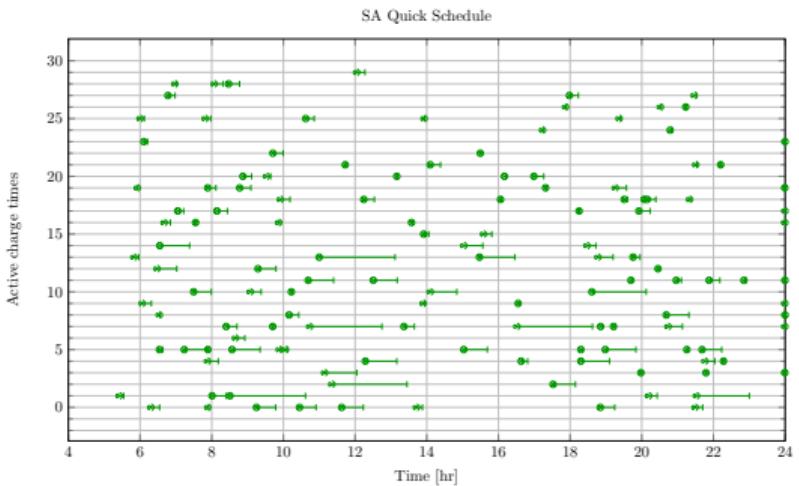
Model	Execution Time [s]	Iteration [s]
MILP	3600	N/A
Quick	2275.25	0.25
Heuristic	3640.4	0.4

- ▶  $T_0 = 99999$
- ▶  $\beta = 0.999$
- ▶  $|t| = 3797$
- ▶  $n_K = 500$

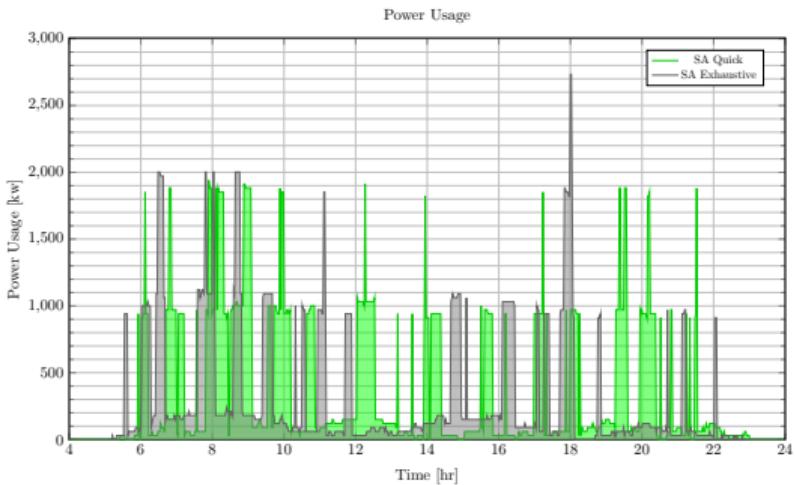
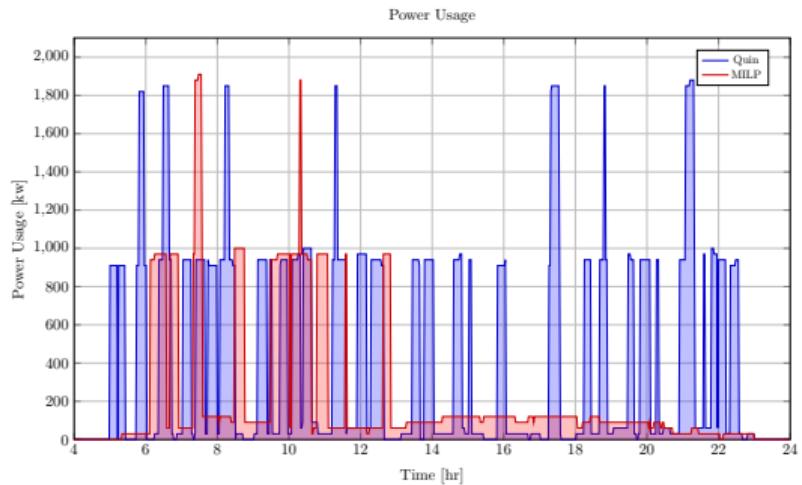
# Schedule



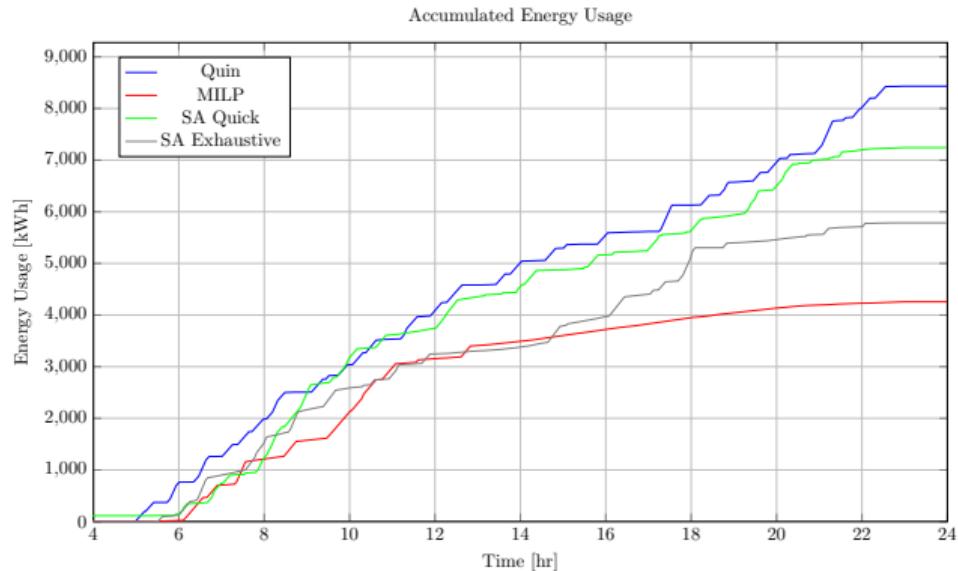
# Schedule



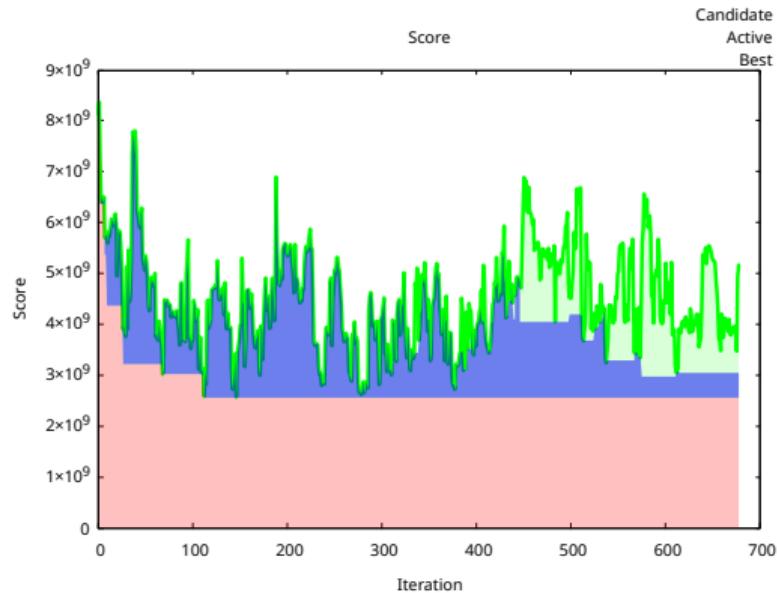
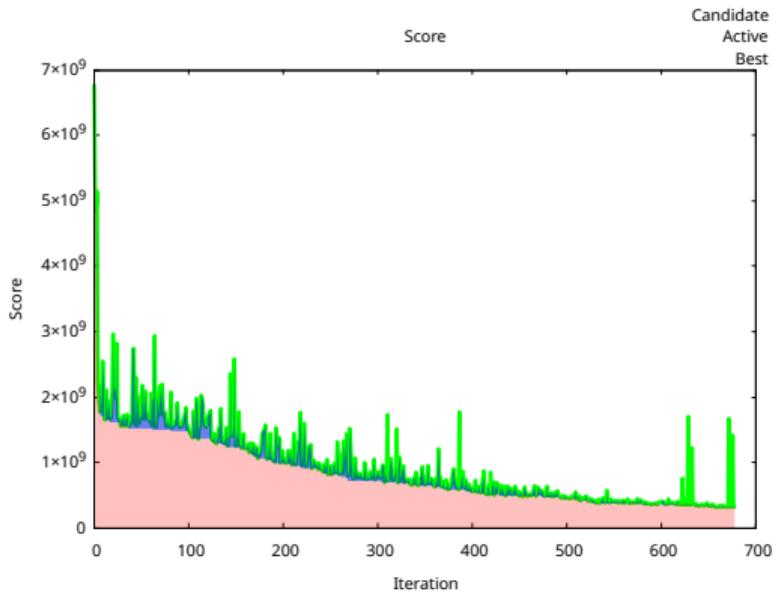
# Power



# Energy



# How To Resolve This Problem?



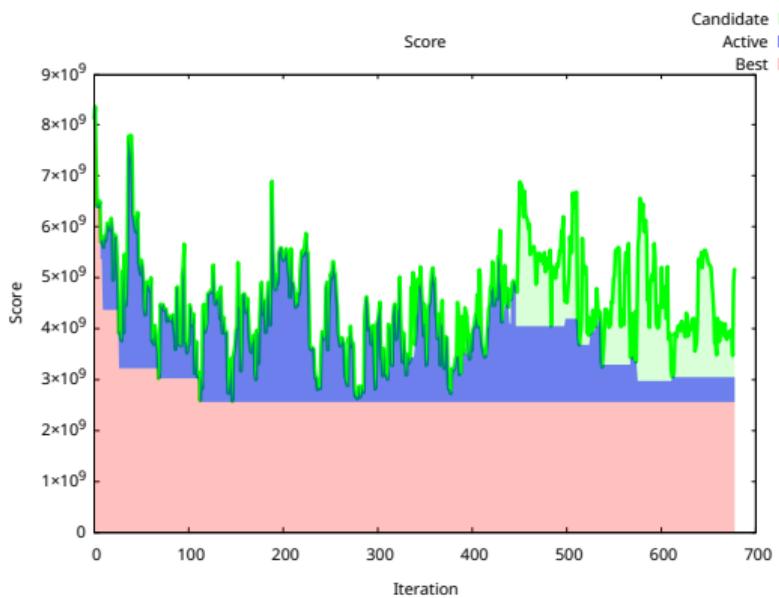
- ▶ Reverse search and weight the visit indices
- ▶ Be more aggressive in exploiting the best solution

- ▶ Candidate solutions diverge
- ▶ Hard time handling “difficult” routes

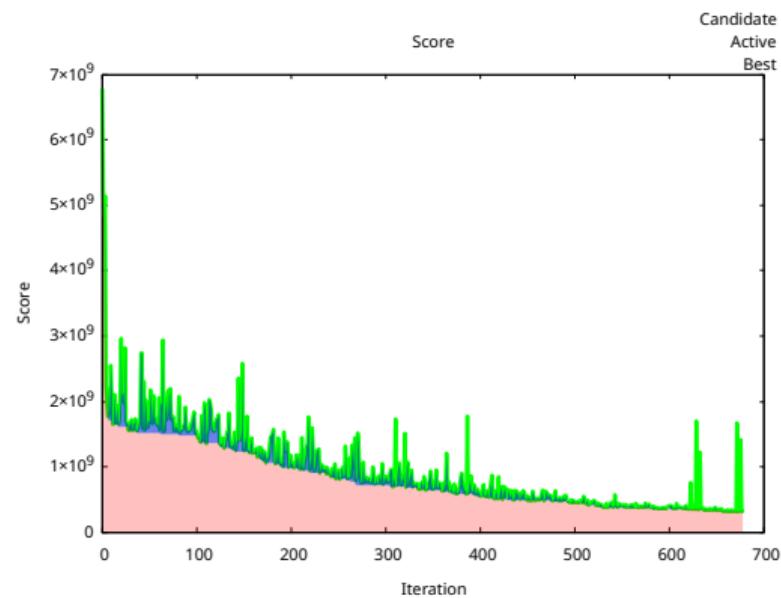
# Score Convergence Comparison



Before Fix



After Fix



## References

-  Mengyuan Duan, Geqi Qi, Wei Guan, Chaoru Lu, and Congcong Gong.  
Reforming mixed operation schedule for electric buses and traditional fuel buses by an optimal framework.  
*IET Intelligent Transport Systems*, 15(10):1287–1303, Jul 2021.
-  Oliver Frendo, Nadine Gaertner, and Heiner Stuckenschmidt.  
Open source algorithm for smart charging of electric vehicle fleets.  
*IEEE Transactions on Industrial Informatics*, 17(9):6014–6022, September 2021.
-  Amra Jahic, Mina Eskander, and Detlef Schulz.  
Preemptive vs. non-preemptive charging schedule for large-scale electric bus depots.  
In *2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*.  
IEEE, September 2019.
-  Daniel Mortensen, Jacob Gunther, Greg Droke, and Justin Whitaker.  
Cost minimization for charging electric bus fleets.  
*World Electric Vehicle Journal*, 14(12):351, December 2023.