A POSITION ALLOCATION PROBLEM APPROACH TO THE BATTERY ELECTRIC BUS CHARGING PROBLEM

Alexander Brown

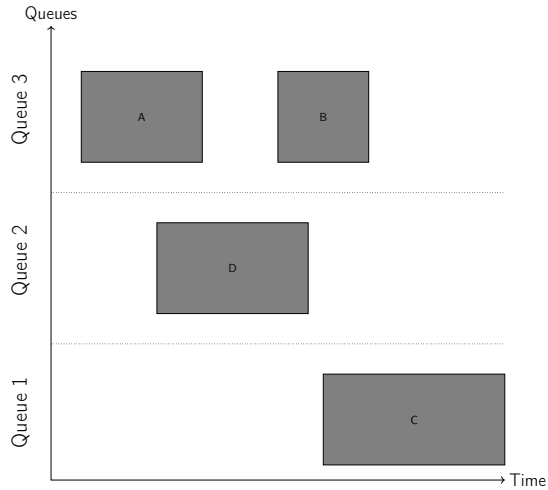
College of Engineering
Utah State University

May 19, 2024

Outline

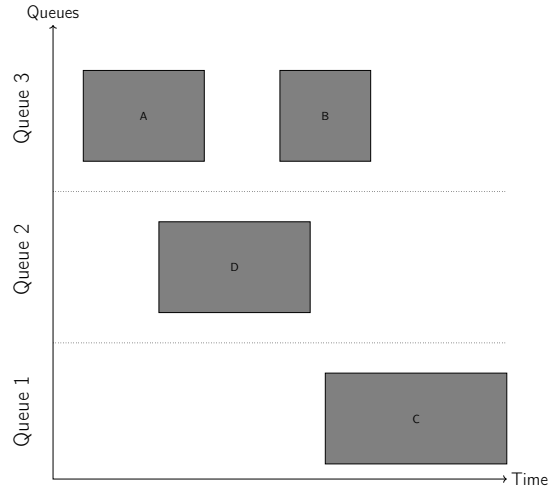
Problem Description

Bus	Start [h]	Stop [h]	Discharge [kWh]
1	1	1.5	50
1	2	2.25	10
2	0	0.75	15
2	1.25	2	10
2	2.5	3	12
3	2	3	20
⋮	⋮	⋮	⋮



Overall Objective

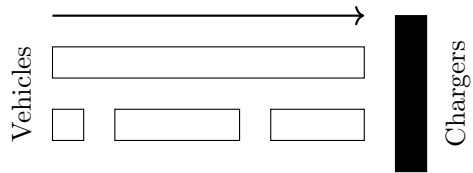
- Create an operationally feasible schedule quickly
- Minimize cost
  - Demand cost
  - Consumption cost
- Battery dynamics
  - Linear
  - Non-linear
- Consider battery health
- Allow partial charging
- Multiple charger types
- Minimize total amount of chargers



Model Objectives - BPAP

BEB Implementation of the PAP (BPAP)

- Minimize cost
  - Consumption cost
- Battery dynamics
  - Linear
- Consider battery health
- Allow partial charging
- Multiple charger types
- Minimize total amount of chargers

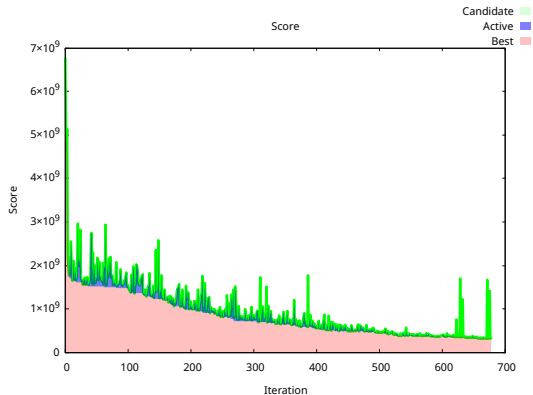


Model Objectives - Linear SA

College of Engineering
UtahStateUniversity

SA PAP with Linear Battery Dynamics

- Create an operationally feasible schedule quickly
- Minimize cost
  - Demand cost
  - Consumption cost
- Linear battery dynamics
- Consider battery health
- Allow partial charging
- Multiple charger types



Brief State Of The Art

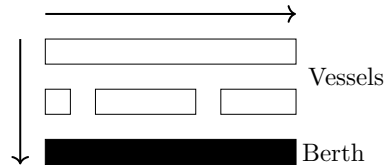
Ref	Consumption	Demand	Linear	Non-linear	Charger Min.	Battery Health
[1]	✓		✓			
[2]	✓	✓	✓			
[3]	✓	✓	✓	✓		
[4]	✓	✓	✓			
[5]	✓	✓	✓			
[6]	✓		✓			
[7]	✓			✓	✓	✓
[8]	✓		✓			
[9]	✓		✓		✓	
BPAP	✓		✓		✓	✓
SA BPAP	✓	✓	✓	✓	*	✓

The Berth Allocation Problem[1]

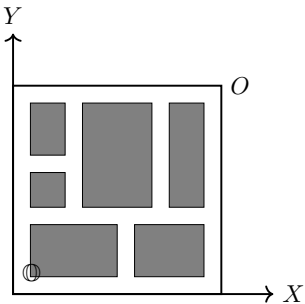
[1] https://www.mdpi.com/2077-1312/11/7/1280

The Berth Allocation Problem

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



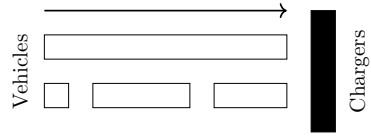
- A variant of the rectangle packing problem
- Solves the problem of optimally assigning incoming vessels to be serviced



The Position Allocation Problem

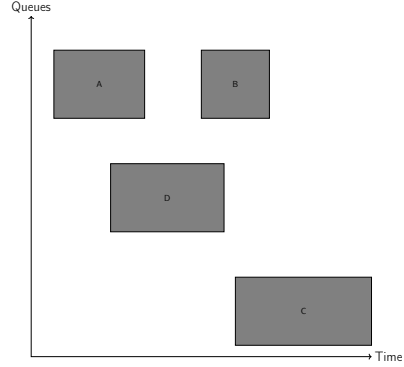
PAP Behavior

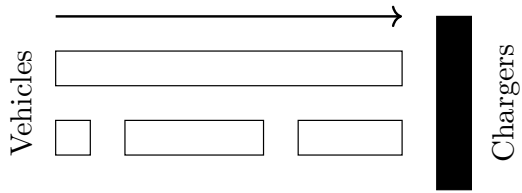
- Service flow is left to right
- Single charger type
- All arrivals are considered unique
- Service times are assumed to be known



Desired Behavior

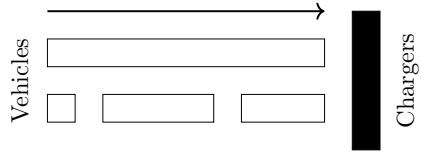
- Discrete charger queues
- Multiple charger types
- Bus can have multiple visits
- Propagate battery SOC across visits

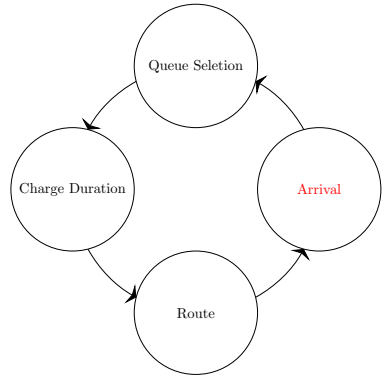




Requirements For BEB Implementation

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





Mixed Integer Linear Program

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

$$u_j - u_i - s_i - (\sigma_{ij} - 1) T \geq 0$$
$$v_j - v_i - (\psi_{ij} - 1) n_Q \geq 1$$
$$\sigma_{ij} + \sigma_{ji} \leq 1$$
$$\psi_{ij} + \psi_{ji} \leq 1$$
$$\sigma_{ij} + \sigma_{ji} + \psi_{ij} + \psi_{ji} \geq 1$$

$$s_i + u_i = d_i$$
$$a_i \leq u_i \leq (T - s_i)$$
$$d_i \leq \tau_i$$

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

$$\eta_i + \sum_{q=1}^{n_Q} g_{iq} r_q - \Delta_i = \eta_{\gamma_i}$$
$$\eta_i + \sum_{q=1}^{n_Q} g_{iq} r_q - \Delta_i \geq \nu_{\Gamma_i} \kappa_{\Gamma_i}$$
$$\eta_i + \sum_{q=1}^{n_Q} g_{iq} r_q \leq \kappa_{\Gamma_i}$$
$$\eta_{\Gamma_b^0} = \alpha_{\Gamma_i} \kappa_{\Gamma_i}$$
$$\eta_{\Gamma_b^f} \geq \beta_{\Gamma_b} \kappa_{\Gamma_b}$$

Queuing Constraints

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

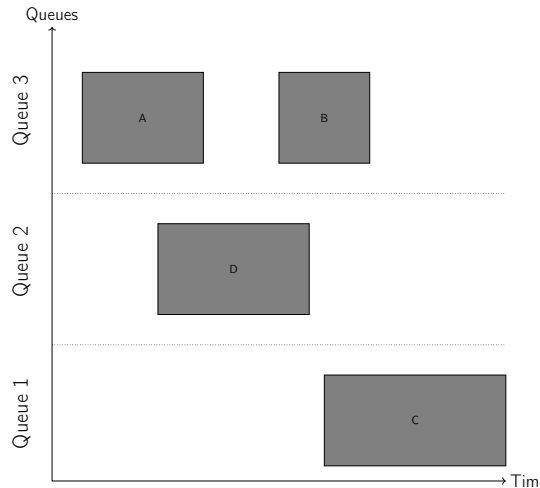
$$v_j - v_i - (\psi_{ij} - 1)n_Q \geq 1$$

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

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

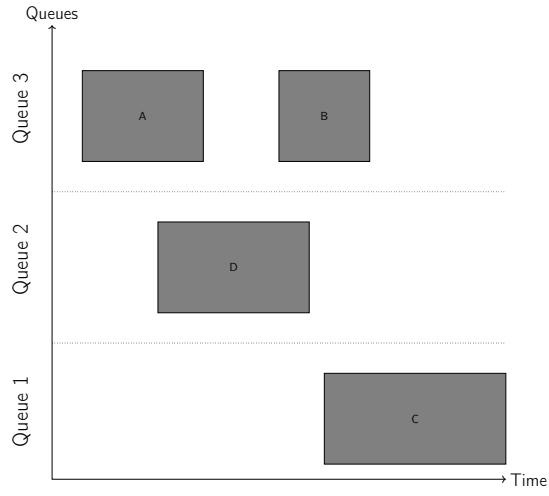
$$\sigma_{ij} + \sigma_{ji} + \psi_{ij} + \psi_{ji} \geq 1$$

- Used to ensure that gray rectangles do not overlap
- σ_{ij} establishes temporal ordering when active
- ψ_{ij} establishes spacial ordering when active

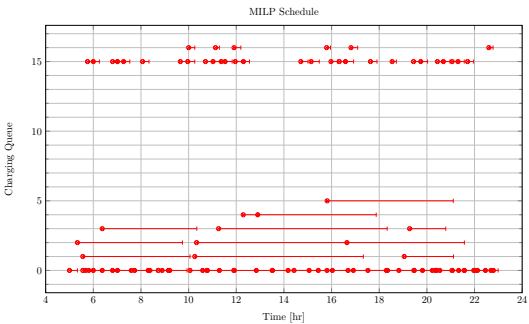


Parameters

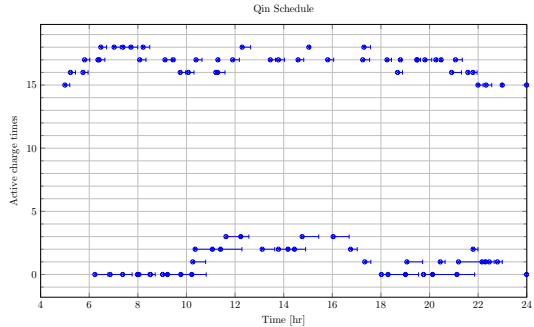
Parameter	Value
Execution	2 hr
T	24 hr
n_V	338
n_A	35
α_i	90%
ν_i	25%
β_i	70%
m_q	1000q



Schedules



- Slow chargers utilized more frequently
- Fast chargers utilized sparingly

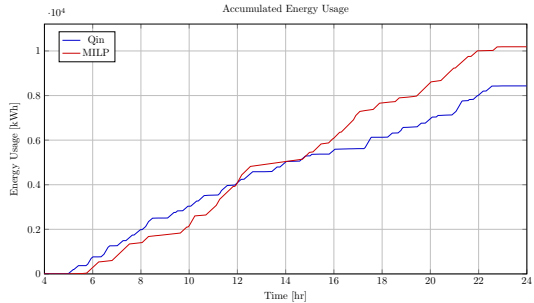


- Slow chargers used for short durations
- Extensive use of fast chargers

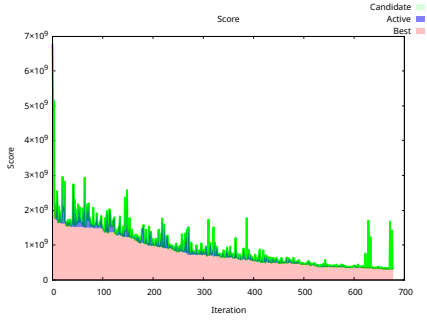
SOC And Energy Use

	BPAP [kWh]	Qin [kWh]
Mean	265.873	355.93
Min	97.04	0.000
Max	388	368.354

- SOC of Qin allowed to drop to 0
- PAP maintained minimum SOC and final SOC



- PAP accumulated energy is larger due to minimum SOC constraints



Objective Function And Constraints

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

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

$$s_i = d_i - u_i$$
$$a_i \leq u_i \leq d_i \leq e_i \leq \mathcal{T}$$

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

- $\epsilon_{q_i} r_{q_i}$: 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

$$u_j - d_i - (\sigma_{ij} - 1)T \geq 0$$
$$q_j - q_i - 1 - (\psi_{ij} - 1)Q \geq 0$$
$$\sigma_{ij} + \sigma_{ji} \leq 1$$
$$\psi_{ij} + \psi_{ji} \leq 1$$
$$\sigma_{ij} + \sigma_{ji} + \psi_{ij} + \psi_{ji} \geq 1$$

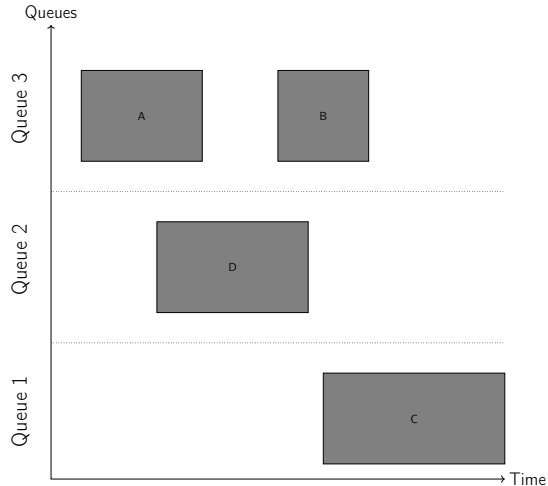
Parameters

NOTES:

- Results are different from what is presented in the thesis
- BPAP model does not take demand cost into consideration

Model	Execution Time [s]
BPAP	1900
Quick	1533
Heuristic	1916

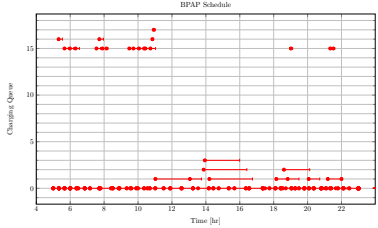
- $T_0 = 90000$
- $\beta = 0.997$
- $|t| = 3797$
- $n_K = 500$



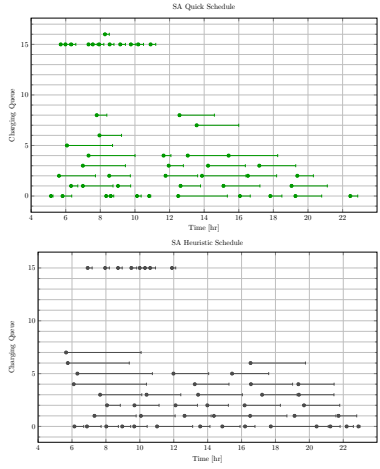
Schedule

College of Engineering
UtahStateUniversity

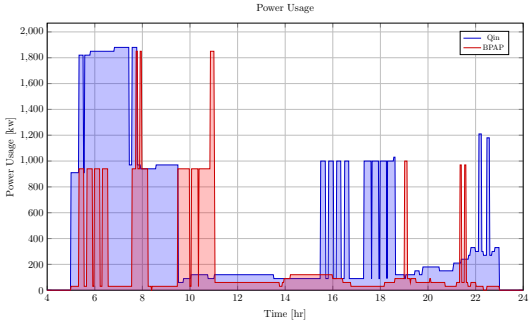
- Qin heavily utilizes fast charger
- BPAP emphasizes slow queues, but utilizes fast chargers readily



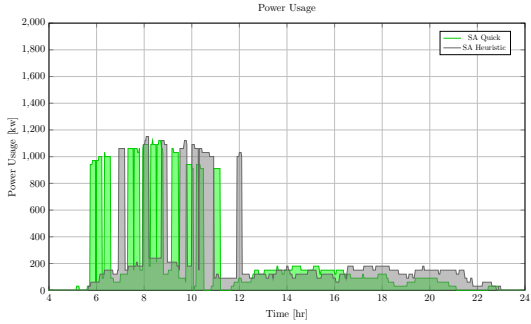
- SA techniques heavily utilize slow charging
- SA techniques utilize fast chargers more sparingly



Power



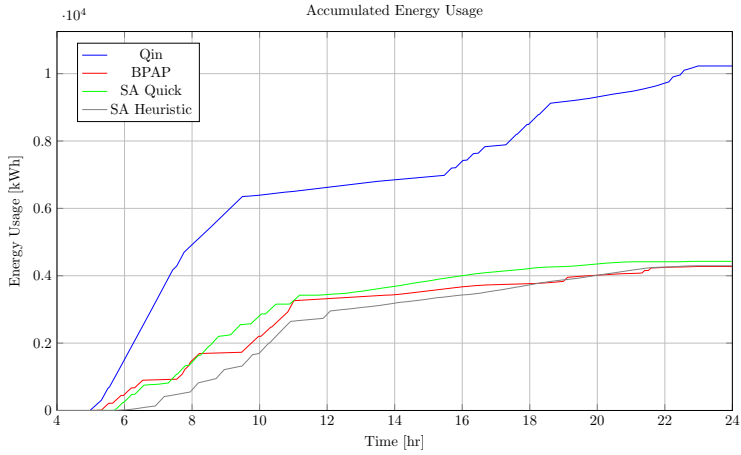
- Heuristic SA maintained the lowest demand peak



- Quick SA peak demand comparable to the PAP peak

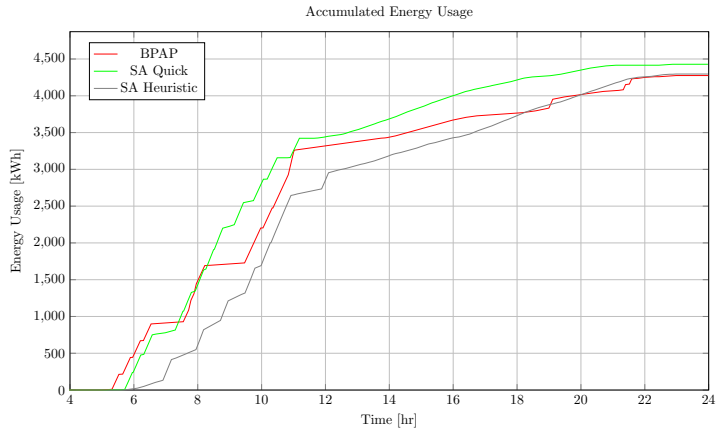
SOC and Energy

- SOC threshold deficits:
  - quick: 2.24 kWh
  - heuristic: 5.74 kWh
- Energy delta:
  - quick: 191.47 kWh
  - heuristic: 58.46 kWh
- Trade off of the lower peak demand



Scores and Energy

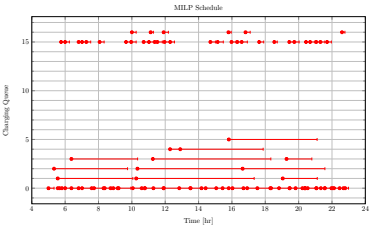
Schedule	Score
BPAP	18,500,000
Qin-Modified	34,578,526
Heuristic	11,673,937
Quick	11,234,577



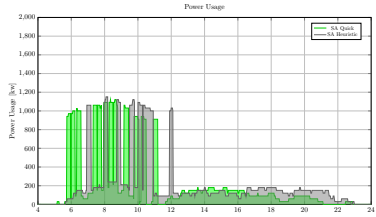
Summary

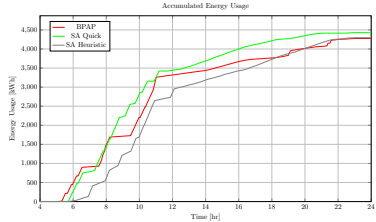
BPAP

	MILP [kWh]	Qin [kWh]
Mean	265.873	355.93
Min	97.04	0.000
Max	388	368.354



SA Techniques





Publications

Droge Greg Brown, Alexander. A simulated annealing approach to the scheduling of battery-eletric bus charging. *Future Transportation*, (4):28, Under Revision. Avaible for download at: https://arxiv.org/submit/5603923.

Droge Greg Brown, Alexander and Jacob Gunther. A position allocation approach to the scheduling of battery-electric bus charging. *Frontiers in Energy Research*, (4):18, Under Revision. Avaible for download at: https://arxiv.org/submit/5603923.

Questions?

Appendix

Mixed Integer Linear Programming

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

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

$$x_j \geq 0 \qquad (j = 1, 2, ..., n)$$

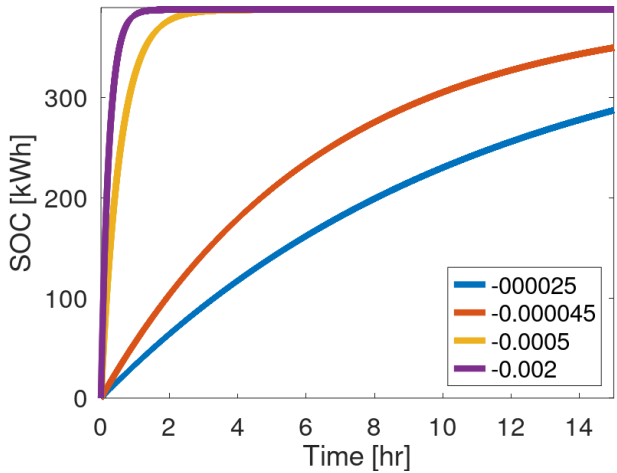
$$y_k \in \mathbb{Z}^+ \qquad (k = 1, 2, ..., n)$$

- 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 with Non-Linear Battery Dynamics

Introduction

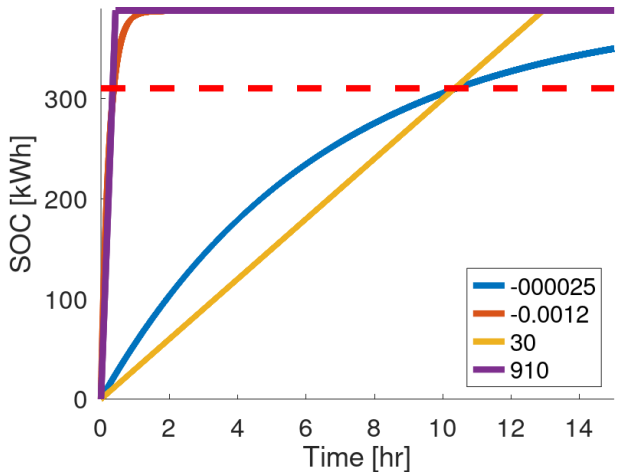
- Higher fidelity in approximating charge at high SOC
- Implemented in SA for simplicity



Non-Linear Battery Dynamics Model

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

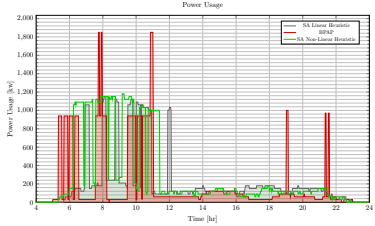
$$\bar{a}_q = e^{a_q dt} \quad \bar{b}_q = e^{a_q dt} - 1$$



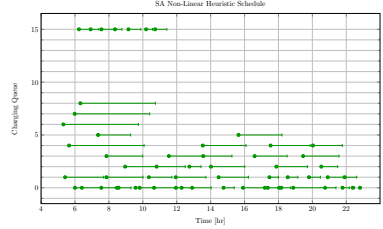
Results

College of Engineering
UtahStateUniversity

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



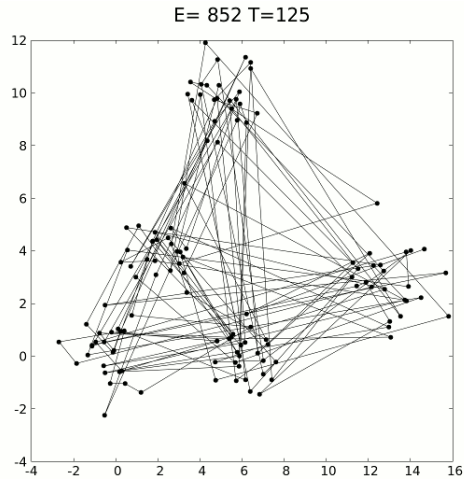
- Similar schedule output, longer fast charger durations due to nonlinear model
- Minimum SOC of 85 kWh



Simulated Annealing

Simulated Annealing[2]

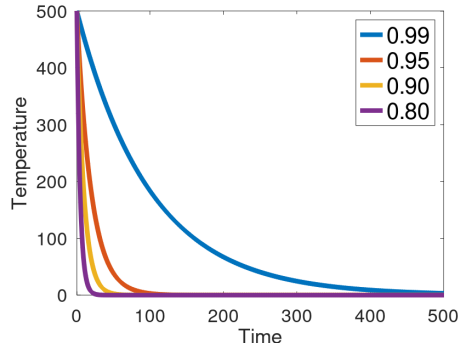
- A probabilistic technique for approximating the global optimum of a given function.
- Often applied to problems that contain many local solutions
- Three key components:
  - Cooling Schedule
  - Acceptance Criteria
  - Generation Mechanisms



[2] https://en.wikipedia.org/wiki/Simulated_annealing

Cooling Schedule

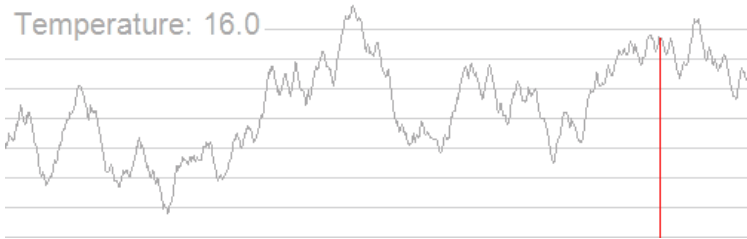
- The cooling equation models the rate at which the temperature decreases over time in the SA process.
- The temperature is high, SA encourages exploration. As the temperature decreases, exploitation is encouraged.

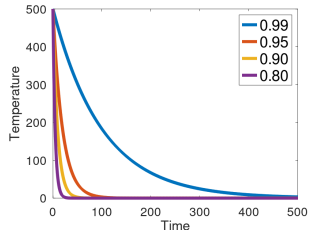


$$t_m = \beta t_{m-1}$$

Acceptance Criteria[3]

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

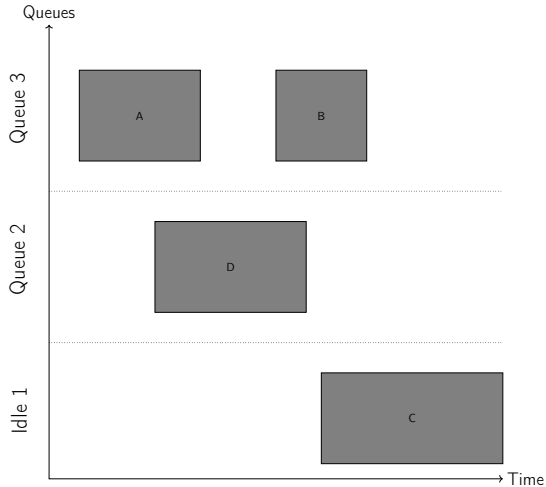




[3] https://en.wikipedia.org/wiki/Simulated_annealing

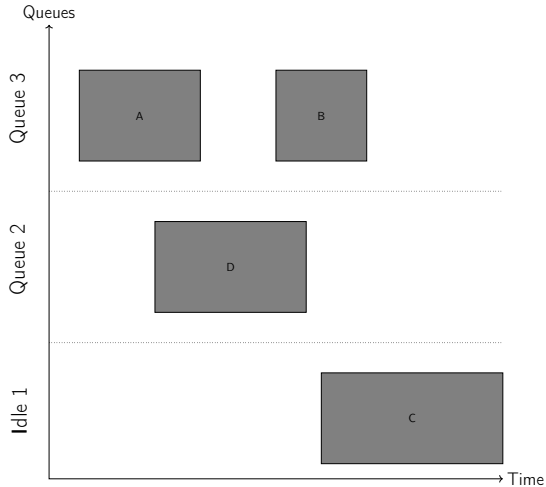
Generation Mechanisms - Primitive Functions

- 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



New Visit

Algorithm 1: New visit algorithm

Algorithm: New Visit

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

begin
  $i \leftarrow \mathbb{S}_i$; /* Extract visit index */
  $\mathbb{I} \leftarrow \mathbb{S}_{\mathbb{I}}$; /* Extract visit tuple */
  $\mathbb{C} \leftarrow \mathbb{S}_{\mathbb{C}}$; /* Extract visit charger availability */
  $a \leftarrow \mathbb{I}_{i.a}$; /* Extract the arrivial time for visit i */
  $e \leftarrow \mathbb{I}_{i.e}$; /* Extract the departure time for visit i */
  $q \leftarrow \mathbb{I}_{i.q}$; /* Extract the current charge queue for visit i */
  $\bar{q} \leftarrow \mathcal{U}_Q$; /* Select a random charging queue with a uniform distribution */
  $C \leftarrow \mathcal{U}_{\mathbb{C}_{\bar{q}}}$; /* Select a random time slice from $\mathbb{C}_{\bar{q}}$ */
  if $(\bar{C}, \bar{u}, \bar{d}) \leftarrow$ FindFreeTime$(C, i, \bar{q}, a, e) \notin \varnothing$ **then** /* If there is time available in C */
    $\bar{I}_{i.q} \leftarrow \bar{q}$; /* Update visit tuple with new charge queue */
    $\bar{I}_{i.u} \leftarrow \bar{u}$; /* Update visit tuple with new inital charge time */
    $\bar{I}_{i.d} \leftarrow \bar{d}$; /* Update visit tuple with new final charge time */
    return $(i, \bar{I}, \bar{C})$ /* Return visit */
  end
  return $(\varnothing)$; /* Return nothing */
end

Algorithm 2: Slide Visit Algorithm

Algorithm: Slide Visit

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

begin
  $i \leftarrow \mathbb{S}_i$; /* Extract visit index */
  $\mathbb{I} \leftarrow \mathbb{S}_{\mathbb{I}}$; /* Extract visit tuple */
  $\mathbb{C} \leftarrow \mathbb{S}_{\mathbb{C}}$; /* Extract visit charger availability */
  $(i, \mathbb{I}, \bar{\mathbb{C}}) \leftarrow$ Purge($\mathbb{S}$); /* Purge visit i from charger availibility matrix */
  $C \leftarrow \bar{C}_{i.q}$; /* Get the time availability of the purged visit */
  /* If there is time available in C */
  if $(\bar{C}, \bar{u}, \bar{d}) \leftarrow$ FindFreeTime(C, i, $\mathbb{I}_{i.q}$, $\mathbb{I}_{i.a}$, $\mathbb{I}_{i.e}$) $\notin \varnothing$ **then**
    $\bar{\mathbb{I}}_{i.u} \leftarrow \bar{u}$; /* Update visit tuple with new inital charge time */
    $\bar{\mathbb{I}}_{i.d} \leftarrow \bar{d}$; /* Update visit tuple with new final charge time */
    return $(i, \bar{\mathbb{I}}, \bar{C})$ /* Return updated visit */
  end
  return $(\varnothing)$; /* Return nothing */
end

Algorithm 3: New Charger Algorithm

Algorithm: New Charger

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

begin
  $i \leftarrow \mathbb{S}_i$; /* Extract visit index */
  $\mathbb{I} \leftarrow \mathbb{S}_{\mathbb{I}}$; /* Extract visit tuple */
  $\mathbb{C} \leftarrow \mathbb{S}_{\mathbb{C}}$; /* Extract visit charger availability */
  $(i, \mathbb{I}, \bar{\mathbb{C}}) \leftarrow$ Purge$(\mathbb{S})$; /* Purge visit i from charger availibility matrix */
  $\bar{q} \leftarrow \mathcal{U}_Q$; /* Select a random charging queue with a uniform distribution */
  if $(\bar{C}, \bar{u}, \bar{d}) \leftarrow$ FindFreeTime$(\bar{\mathbb{C}}_{i.q}, i, \bar{q}, \mathbb{I}_{i.a}, \mathbb{I}_{i.e}) \notin \varnothing$ **then** /* If there is time available in C_q */
    /* Return visit, note u and d are the original inital/final charge times. */
    $\bar{I}_{i.q} \leftarrow \bar{q}$; /* Update visit tuple with new charge queue */
    return $(i, \bar{I}, \bar{\mathbb{C}})$
  end
  return $(\varnothing)$; /* Return nothing */
end

Wait

Algorithm 4: Wait algorithm

Algorithm: Wait

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

begin
  $(i, \mathbb{I}, \bar{\mathbb{C}}) \leftarrow$Purge$(\mathbb{S})$; /* Purge visit i from charger availibility matrix */
  /* Update the charger availability matrix for wait queue $\bar{\mathbb{C}}_{i.q_i}$ */
  $\bar{\mathbb{C}}'_{\mathbb{I}_{i.\Gamma_i}} \leftarrow \mathbb{C}' \cup \{[\mathbb{I}_{i.a}, \mathbb{I}_{i.e}]\}$;
  $\bar{\mathbb{I}}_{i.q} \leftarrow \mathbb{I}_{i.b}$; /* Reassign bus to idle queue */
  $\bar{\mathbb{I}}_{i.u} \leftarrow \mathbb{I}_{i.a}$; /* Set initial charge time to the arrival time */
  $\bar{\mathbb{I}}_{i.d} \leftarrow \mathbb{I}_{i.e}$; /* Set the final charge time to the departure time */
  return $(i, \bar{\mathbb{I}}, \bar{\mathbb{C}})$ /* Return visit */
end

New Window

Algorithm 5: New window algorithm

Algorithm: New Window

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

```
1 begin
2     S̄ ← Wait(S);                          /* Assign visit to its respective idle queue */
3     if S̄ ← NewVisit(S̄) ∉ ∅ then           /* Add visit i back in randomly */
4         return S̄                           /* Return visit */
5     end
6     return (∅);                           /* Return nothing */
7 end
```

Charge Schedule Generation

Algorithm 6: Charge schedule generation algorithm

Algorithm: Candidate Solution Generator

Input: $(\mathbb{I}, \mathbb{C})$
Output: $(\bar{\mathbb{I}}, \bar{\mathbb{C}})$

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

Algorithm 7: Perturb schedule algorithm

Algorithm: Perturb Schedule

Input: $(\mathbb{I}, \mathbb{C})$
Output: $(\bar{\mathbb{I}}, \bar{\mathbb{C}})$

begin
  $p \leftarrow [false; n_A]$; /* Create vector of booleans to track priority status */
  $y^v \leftarrow [1.0; n_V]$; /* Create weight vector for index selection */
  /* Loop through the visits in reverse order */
  for $i \leftarrow |\mathbb{I}|$ *TO* 1 **do**
    /* Check whether the current visit is part of a priority route */
    if $p_{\mathbb{I}_{i.b}} = true$ **then**
      $y^v_{\mathbb{I}_i} = y^v_{\mathbb{I}_{i.\xi}}$; /* Propagate the priority level to previous visit */
    end
    /* Prioritize if the current visit's SOC falls below the allowed threshold */
    else if $\mathbb{I}_{i.\eta} \leq \nu_{\mathbb{I}_{i.b}} \kappa_{\mathbb{I}_{i.b}}$ **then**
      $p_{\mathbb{I}_{i.b}} = true$; /* Indicate the current BEB's routes are to be prioritized */
      $y^v_{\mathbb{I}_i} = \kappa_{\mathbb{I}_{i.b}} + \kappa_{\mathbb{I}_{i.b}}(\nu_{\mathbb{I}_{i.b}} \kappa_{\mathbb{I}_{i.b}} - \mathbb{I}_{i.\eta})$; /* Calculate the weight of the current visit */
    end
  end
  $i \leftarrow \mathcal{W}^{y^v}_{\mathbb{I}}$; /* Select an index with a weighted distribution */
  $y^p \leftarrow [y^p_{\mathbf{1}}, y^p_{\mathbf{2}}, \ldots]$; /* Define the weight of each primitive generator */
  /* Select a primitve generating function with weighted distribution */
  PrimitiveGeneratingFunction $\leftarrow \mathcal{W}^{y^p}_{[\mathbf{1}, n_G]}$;
  $(i, \bar{\mathbb{I}}, \bar{\mathbb{C}}) \leftarrow$ PrimitiveGeneratingFunction$((i, \mathbb{I}, \mathbb{C}))$; /* Excecute the generator function */
  return $(\bar{\mathbb{I}}, \bar{\mathbb{C}})$
end

Thesis Results

Parameters

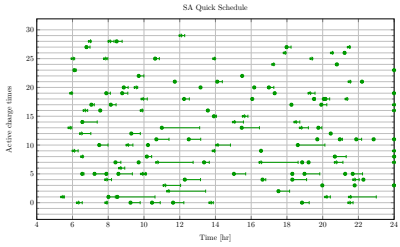
Model	Execution Time [s]	Iteration [s]
MILP	3600	N/A
Quick	2275.25	0.25
Heuristic	3640.4	0.4

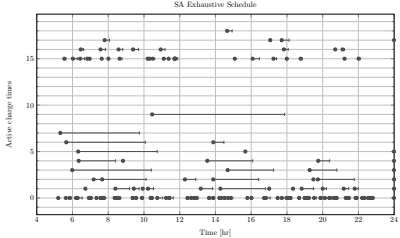
- $T_0 = 99999$
- $\beta = 0.999$
- $|t| = 3797$
- $n_K = 500$

Schedule

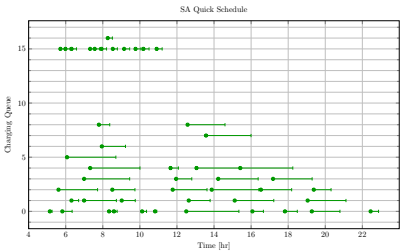
College of Engineering
UtahStateUniversity

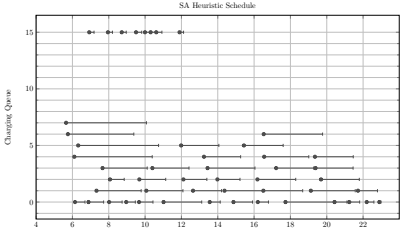
Before





After

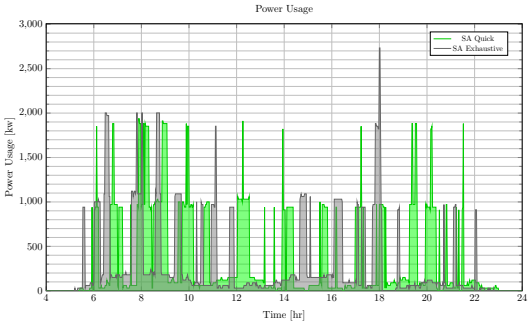




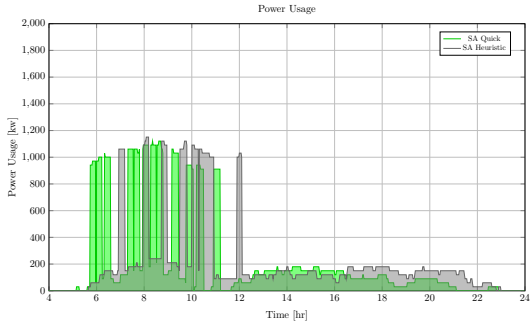
Power

College of Engineering
UtahStateUniversity

Before



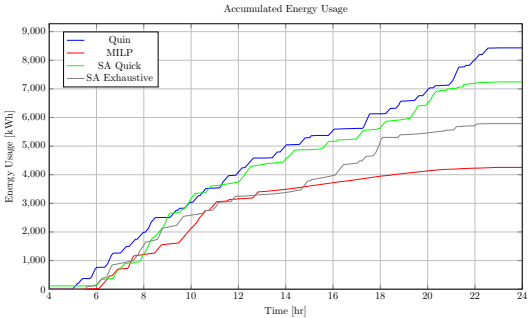
After



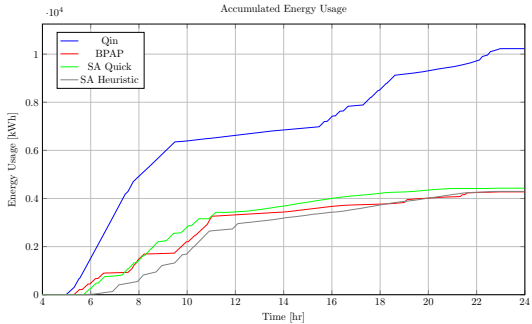
Energy

College of Engineering
UtahStateUniversity

Before



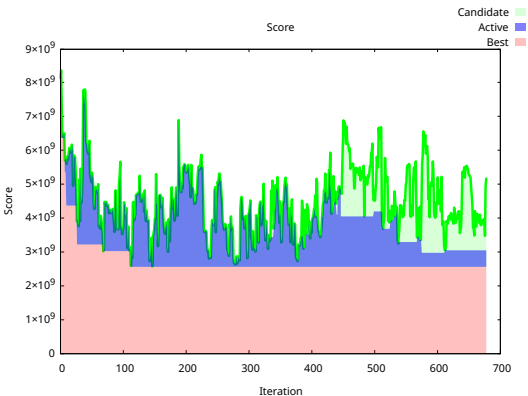
After



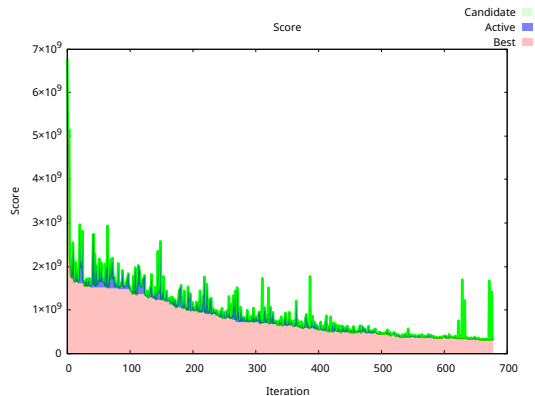
Score Convergence Comparison

College of Engineering
UtahStateUniversity

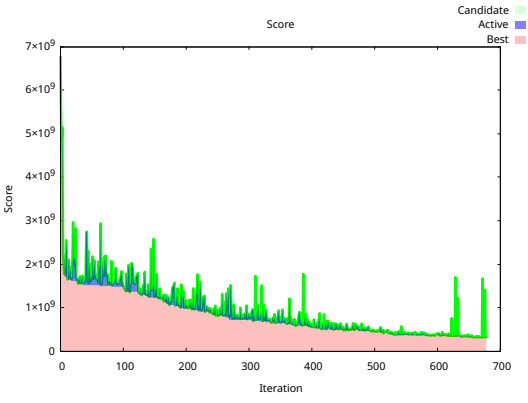
Before Fix



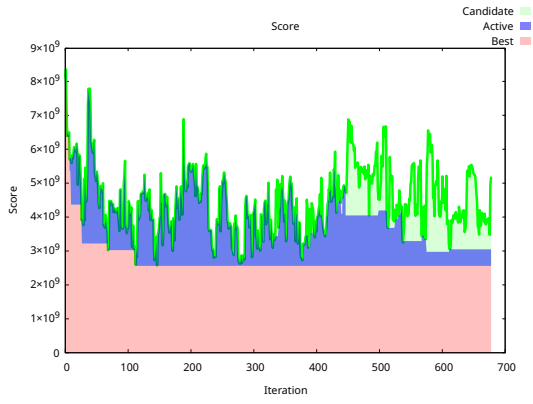
After Fix



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

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 Droge, and Justin Whitaker.
Cost minimization for charging electric bus fleets.
World Electric Vehicle Journal, 14(12):351, December 2023.