A SIMULATED ANNEALING APPROACH TO THE BATTERY ELECTRIC BUS CHARGING PROBLEM

Alexander Brown

College of Engineering
Utah State University

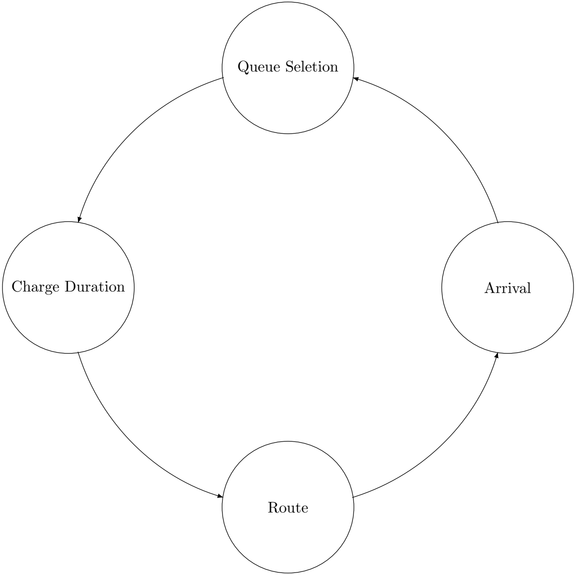
May 5, 2024

Outline

Section 1

Introduction

Problem Description



Mixed Integer Linear Programming

$$
\begin{aligned}
\max\ J &= \sum_j c_j x_j + \sum_k d_k y_k \\
\text{subject to } & \sum_j a_{ij} x_j + \sum_k g_{ik} y_k \le b_i \qquad (i = 1, 2, \ldots, m) \\
& x_j \ge 0 \qquad (j = 1, 2, \ldots, n) \\
& y_k \in \mathbb{Z}^+ \qquad (k = 1, 2, \ldots, n)
\end{aligned}
$$

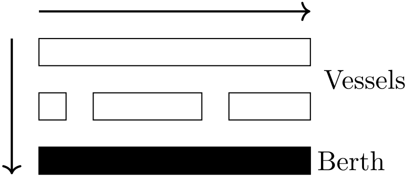
- J: Objective function
- $x_j \in \mathbb{R}$ and $y_k \in \mathbb{Z}^+$: Decision Variables
- $c_j, d_k, a_{ij}, g_{ik}, b_i \in \mathbb{R}$: Parameters

The Berth Allocation Problem[1]

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

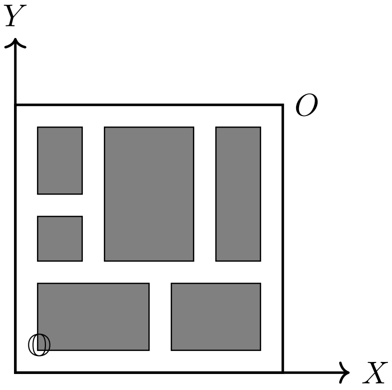
The Berth Allocation Problem

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



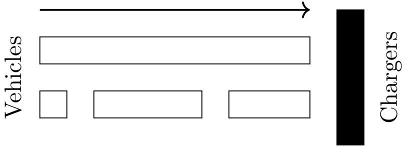
The Berth Allocation Problem

- A variant of the rectangle packing problem
- Solves the problem of optimally assigning incoming vessels to be serviced
  - $\mathbb{O}$: Spatiotemporal allocations for vessels
  - O: Time horizon and berthing space



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



Section 2

The Position Allocation Problem Approach With Linear Battery Dynamic

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

Objective Function

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

- $w_{iq} m_q$: Assignment cost
- $g_{iq} \epsilon_q$: Consumption cost

Packing Constraints

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

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

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

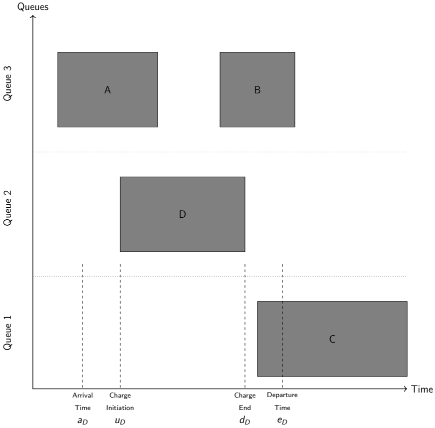
$$\sigma_{ij} + \sigma_{ji} \leq 1$$

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

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

Packing Constraints

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



Packing Constraints

- Calculates the charge duration
- Ensures the arrival time is before the initial charge time
- The initial charge time must be early enough as to not go over the time horizon
- The detach time must be before the departure time

$$s_i + u_i = d_i$$

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

$$d_i \leq \tau_i$$

Linear Battery Dynamic Constraints

- Calculates the charge for the next visit
- Ensures the current charge is above the minimum charge threshold
- Ensures the current charge is below the battery capacity

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

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

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

Bilinear Linearization Constraints

- Linearization of bilinear terms

$$g_{iq} = \begin{cases} s_i & w_{iq} = 1 \\ 0 & w_{iq} = 0 \end{cases}.$$

$$s_i - (1 - w_{iq})M \leq g_{iq}$$

$$s_i \geq g_{iq}$$

$$Mw_{iq} \geq g_{iq}$$

$$0 \leq g_{iq}$$

Charging Queue Constraints

- Ensure only one queue is selected per visit
- Convert vector representation of queue selection to an integer

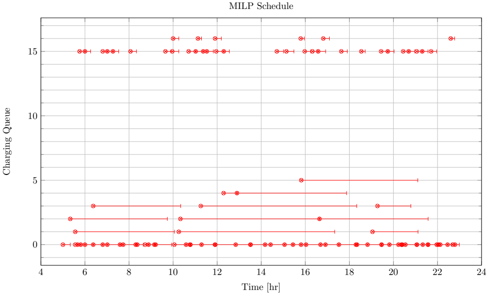
$$\sum_{q=1}^{n_Q} w_{iq} = 1$$

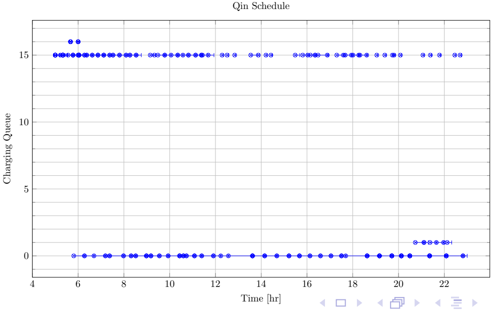
$$v_i = \sum_{q=1}^{n_Q} q w_{iq}$$

Results

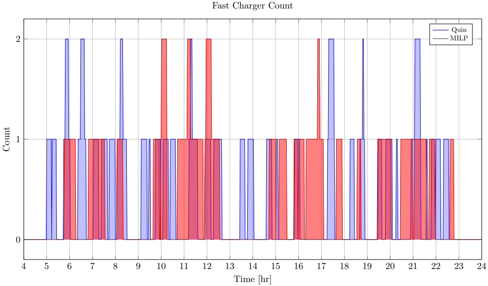
- Executed for 7200 seconds (2 hours)
- $T = 24$
- $n_V = 338$
- $n_A = 35$
- $\alpha_i = 90\%;\ \nu_i = 20\%;\ \beta_i = 70\%$
- $\forall q \in \{n_B + 1, n_B + 2, ..., n_B + n_C\}; m_q = 1000q$

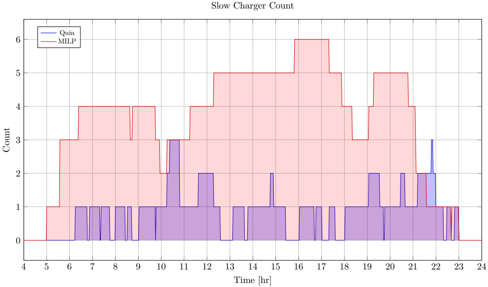
Results


MILP Schedule
Charging Queue
Time [hr]
0
5
10
15
4
6
8
10
12
14
16
18
20
22
24


Qin Schedule
Charging Queue
Time [hr]
0
5
10
15
4
6
8
10
12
14
16
18
20
22

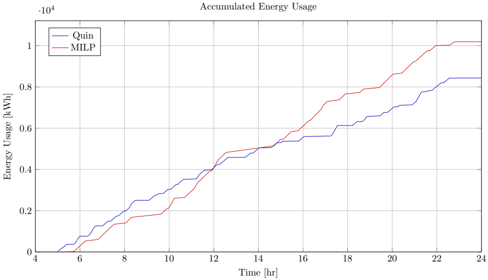
Results

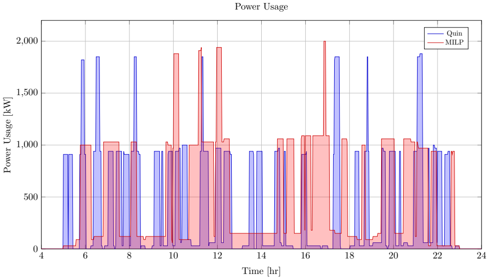




College of Engineering
UtahStateUniversity

Results


Accumulated Energy Usage
·10^4
Quin
MILP
Energy Usage [kWh]
Time [hr]
0
0.2
0.4
0.6
0.8
1
4
6
8
10
12
14
16
18
20
22
24


Power Usage
Quin
MILP
Power Usage [kW]
Time [hr]
0
500
1,000
1,500
2,000
4
6
8
10
12
14
16
18
20
22
24

Section 3

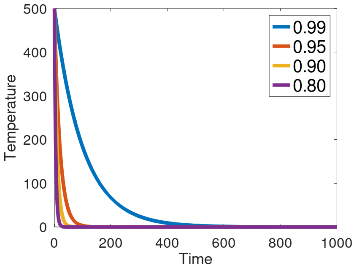
The Simulated Annealing Approach With Linear Battery Dynamics

Simulated Annealing

- Named after its analogized process where a crystalline solid is heated then allowed to cool at a slow rate until it achieves its most regular possible crystal lattice configuration
- The algorithm is often applied to problems that contain many local solutions as it employs a stochastic approach that explores the solution space for an approximate global optimum.
- Within the SA process there are three key components
  - Cooling Schedule
  - Acceptance Criteria
  - Generation Mechanisms

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

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

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

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 */`
  $a \leftarrow \mathbb{I}_{i.a}$; `/* 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 */`
  $C \leftarrow \mathcal{U}_{\mathbb{C}_q}$; `/* Select a random time slice from` $\mathbb{C}_q$ `*/`
  if $(\bar{C}, \bar{u}, \bar{d}) \leftarrow$ `findFreeTime(`C, i, q, a, e`)` $\notin \varnothing$ **then** `/* If there is time available in` C_q^j `*/`
    return $(i, (\bar{q}, \bar{u}, \bar{d}), \bar{C})$ `/* Return visit */`
  end
  return $(\varnothing)$; `/* Return nothing */`
end

Slide Visit

Algorithm 2: Slide Visit Algorithm

Algorithm: Slide Visit

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 */`
- $C \leftarrow \bar{C}_{i.q_i}$; `/* 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, $\mathbb{S}_i$, $\mathbb{I}_q$, $\mathbb{I}_{i.a}, \mathbb{I}_{i.e}$`)` $\notin \varnothing$ **then**
  - **return** $(i, \mathbb{I}, (\mathbb{I}_{i.q_i}, \bar{u}, \bar{d}), \bar{C})$ `/* Return updated visit */`
- **end**
- **return** $(\varnothing)$; `/* Return nothing */`

end

New Charger

Algorithm 3: New Charger Algorithm

Algorithm: New Charger

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 */`
- $q \leftarrow \mathcal{U}_Q$; `/* Select a random charging queue with a uniform distribution */`
- **if** $(\bar{C}, \bar{u}, \bar{d}) \leftarrow$ `findFreeTime(`$\bar{\mathbb{C}}_{i.q}$, $\mathbb{S}_i$, $\mathbb{I}_q$, $\mathbb{I}_{i.a}, \mathbb{I}_{i.e}$`)` $\notin \varnothing$ **then** `/* If there is time available in` C_q `*/`
  - `/* Return visit, note` u `and` d `are the original inital/final charge times. */`
  - **return** $(i, \mathbb{I}, (q, \mathbb{I}_{i.u}, \mathbb{I}_{i.d}), \bar{\mathbb{C}})$
- **end**
- **return** $(\varnothing)$; `/* Return nothing */`

end

Wait

Algorithm 4: Wait algorithm

Algorithm: Wait

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

```
begin
    (i, 𝕀, C̄) ← Purge(𝕊);        /* Purge visit i from charger availibility
                                    matrix */
    C̄'_{𝕀_{i.Γ_i}} ← ℂ' ∪ {[𝕀_{i.a}, 𝕀_{i.e}]};  /* Update the charger availability matrix
                                    for wait queue C̄_{i.q_i} */
    return (i, 𝕀, (𝕀_{i.b}, 𝕀_{i.a}, 𝕀_{i.e}), C̄)      /* Return visit */
end
```

New Window

Algorithm 5: New window algorithm

Algorithm: New Window

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

begin
    $\bar{\mathbb{S}} \leftarrow$ `Wait(`$\mathbb{S}$`);`   /* Assign visit to its respective idle queue */
    if $\bar{\bar{\mathbb{S}}} \leftarrow$ `NewVisit(`$\bar{\mathbb{S}}$`)` $\notin \varnothing$ **then**   /* Add visit i back in randomly */
        return $\bar{\bar{\mathbb{S}}}$ /* Return visit */
    end
    return $(\varnothing)$;   /* Return nothing */
end

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

Charge Schedule Generation

Algorithm 6: Charge schedule generation algorithm

Algorithm: Candidate Solution Generator

Input: $\mathbb{S}$

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

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 $(0, \bar{\mathbb{I}}, \bar{\mathbb{C}})$

end

Perturb Schedule

Algorithm 7: Perturb schedule algorithm

Algorithm: Perturb Schedule

Input: $\mathbb{S}$

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

begin

  $p \leftarrow [false; n_A]$; /* Create vector to track priority routes */

  $y^i \leftarrow [1.0; n_V]$; /* Create weight vector for index selection */

  /* Loop through the visits in reverse order */

  foreach $\mathbb{I}_i \leftarrow \mathbb{I}_{|\mathbb{I}|}$ *TO* $\mathbb{I}_1$ **do**

    /* If the current visit is part of a priority route */

    if $p_{\mathbb{I}_{i.b}} = true$ **then**

      $y^i_{\mathbb{I}_i} = y^i_{\mathbb{I}_{i.\xi}}$;

    end

    /* Else if the current visit's SOC does 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 */

      $y^i_{\mathbb{I}_i} = \kappa_{\mathbb{I}_{i.b}} (\nu_{\mathbb{I}_{i.b}} \kappa_{\mathbb{I}_{i.b}} - \mathbb{I}_{i.\eta})$; /* Calculate the weight of the current visit */

    end

  end

  $\mathbb{I}_i \leftarrow \mathcal{W}_{\mathbb{I}}^{y^i}$; /* Select an index with a weighted distribution */

  $i \leftarrow \mathbb{I}_i$; /* Extract visit index */

  $y^p \leftarrow [y^p_1, y^p_2, \ldots]$; /* Define the weight of each primitive generator */

  $PGF \leftarrow \mathcal{W}^{y^p}_{[1, n_G]}$; /* Select a generator function with weighted distribution */

  $\bar{\mathbb{S}} \leftarrow \mathtt{PGF}((i, \mathbb{I}, \mathbb{C}))$; /* Excecute the generator function */

  return $(0, \bar{\mathbb{I}}, \bar{\mathbb{C}})$

end

College of Engineering
UtahStateUniversity

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
  - $p_{T_p}[h] = \frac{1}{T_p} \sum_{h-\frac{T_p}{dt}+1}^{h} p_h$
  - $p_d = \max(p_{fix}, p_{max})$
  - $p_{max} = \max_{h \in H} p_{T_p}[h]$
- $\epsilon_{q_i} r_{q_i}$: 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

College of Engineering
UtahStateUniversity

Constraints

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

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

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

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

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

$$s_i = d_i - u_i$$

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

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

$$a_i \leq u_i \leq d_i \leq e_i \leq \mathcal{T}$$

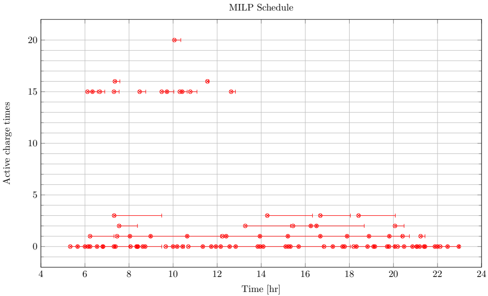
College of Engineering
UtahStateUniversity

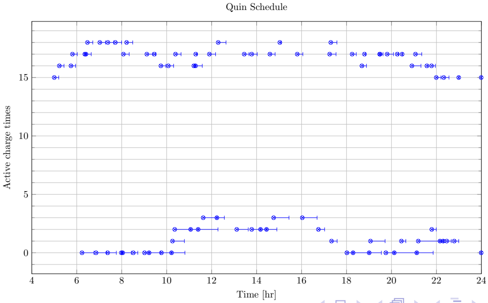
Results - What Is In The Thesis

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

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

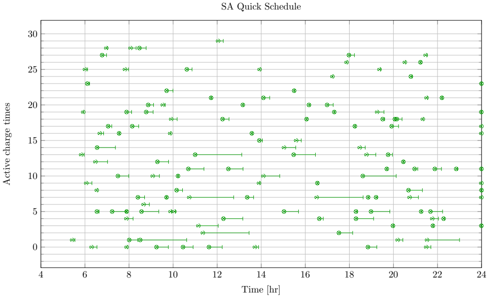
Schedule- What Is In The Thesis

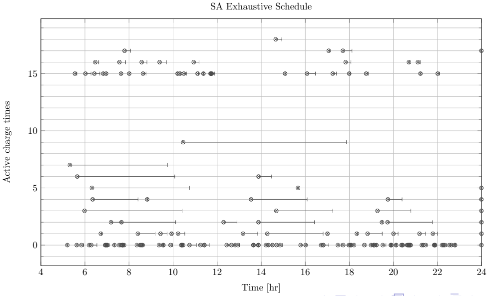




College of Engineering
UtahStateUniversity

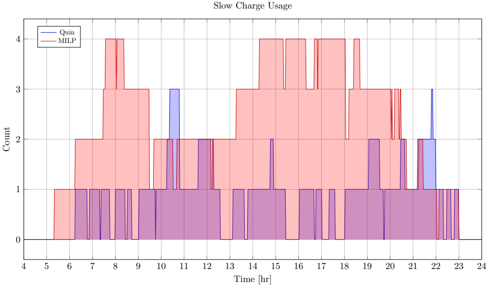
Schedule - What Is In The Thesis

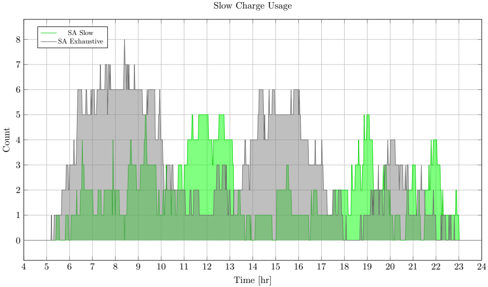




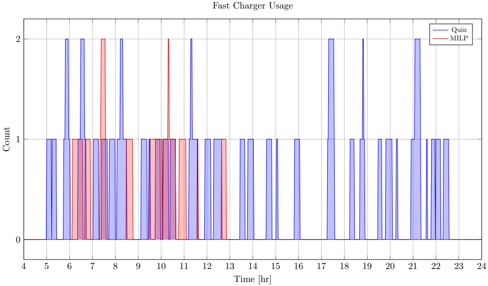
College of Engineering
UtahStateUniversity

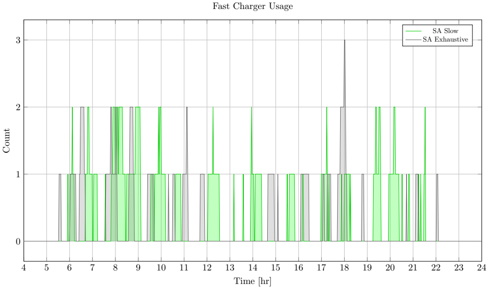
Charger Count - What Is In The Thesis



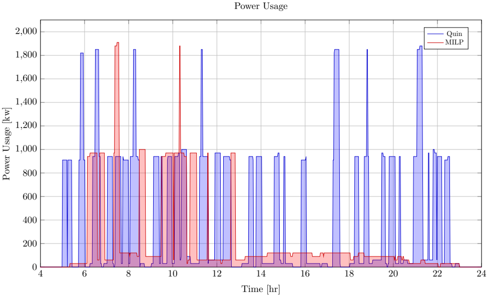


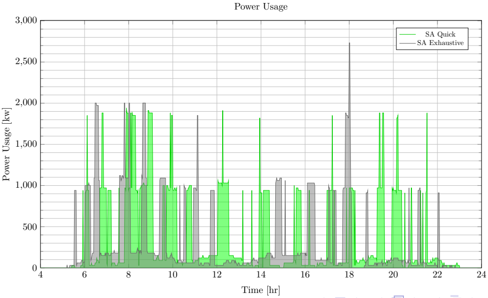
Charger Count - What Is In The Thesis





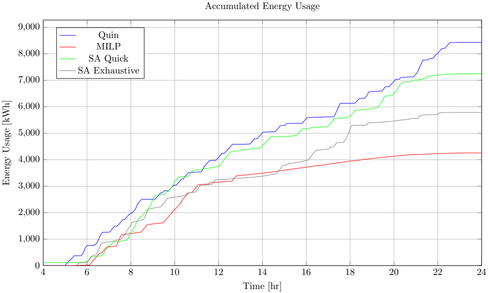
Power - What Is In The Thesis



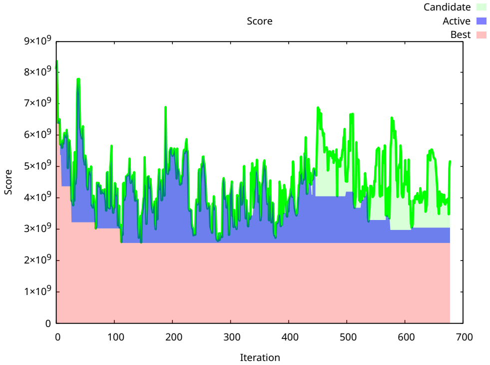


College of Engineering
UtahStateUniversity

Energy - What Is In The Thesis

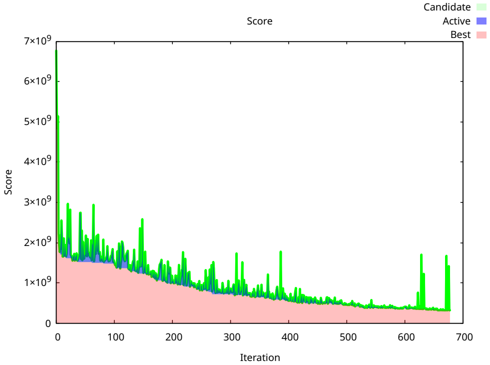


What Happened?



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

How To Resolve This Problem?

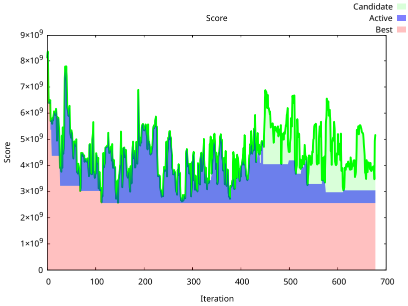


- Reverse search and weight the visit indices
- Be more aggressive in exploiting the best solution

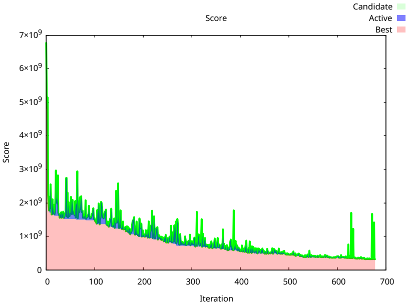
College of Engineering
UtahStateUniversity

Score Convergence Comparison

Before Fix



After Fix

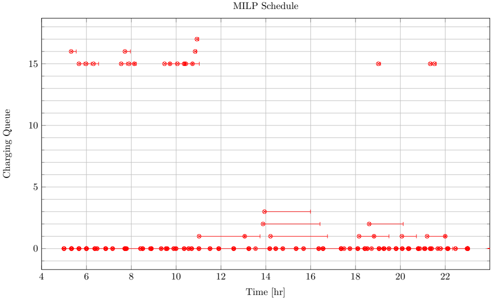


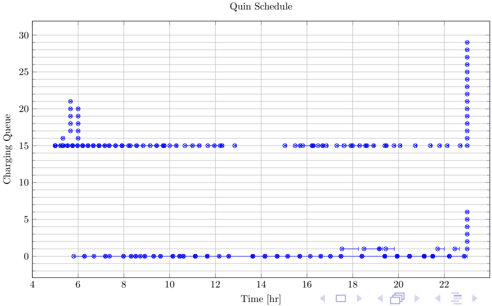
Results - What Is Not In The Thesis

Model	Execution Time [s]	Iteration [s]
MILP	1900	N/A
Quick	1532.8	0.4
Heuristic	1916	0.5

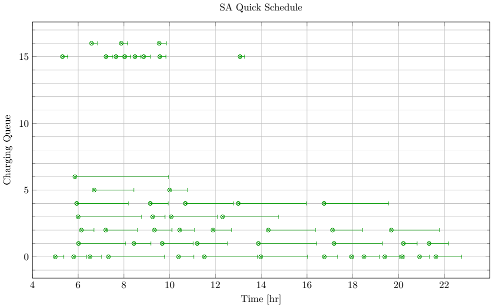
- $T_0 = 90000$
- $\beta = 0.997$
- $|t| = 3797$
- $n_K = 500$

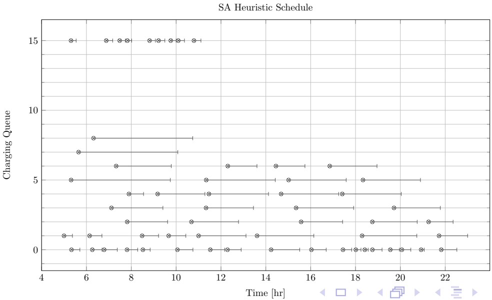
Schedule - What Is Not In The Thesis



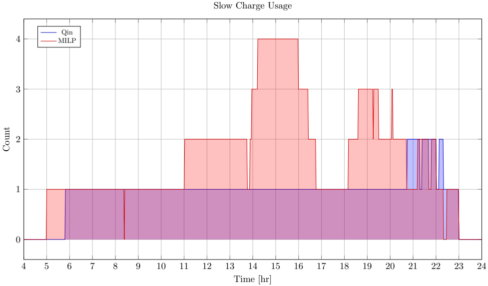


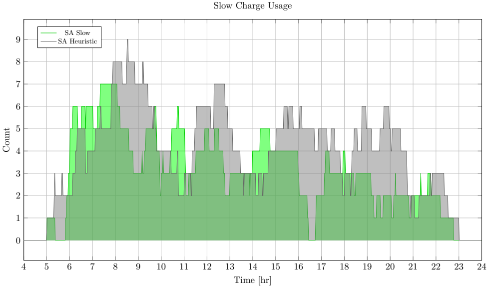
Schedule - What Is Not In The Thesis



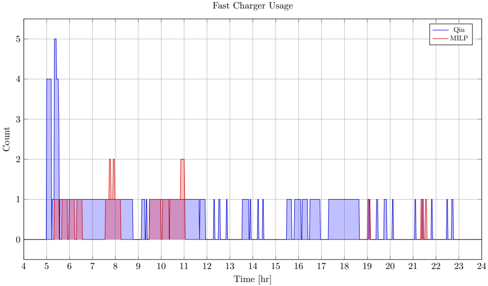


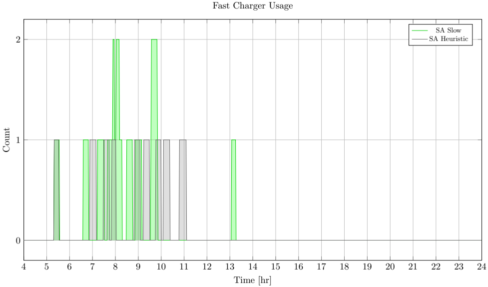
Charger Count - What Is Not In The Thesis





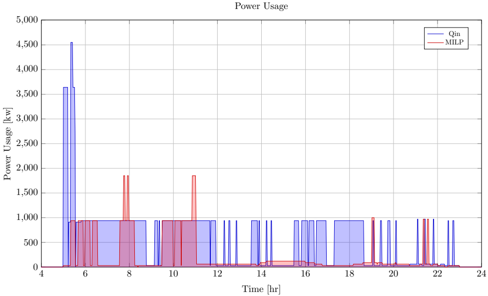
Charger Count - What Is Not In The Thesis

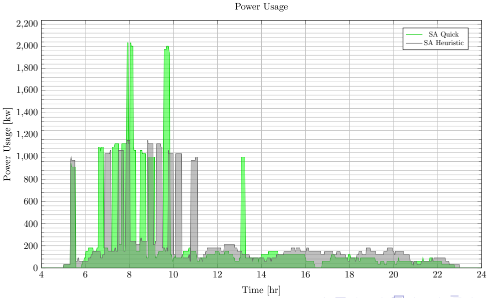




College of Engineering
UtahStateUniversity

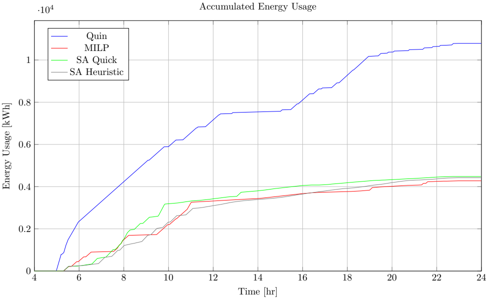
Power - What Is Not In The Thesis





College of Engineering
UtahStateUniversity

Energy - What Is Not In The Thesis



Section 4

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

- Show function
- Show plots

Results

- Figures!