A logo for a university

Description automatically generated

**CAPACITATED EV ROUTING MODEL FOR STUDENT TRANSPORTATION**

**AT SAN JOSÉ STATE UNIVERSITY**

**GROUP 09**

Kavana Pullaganti Sathyanarayana

Mahima Korad Siddalinga Prasad

Yugandhara Uday Kadam

**SAN JOSE STATE UNIVERSITY**

ISE 247: Logistics for Supply Chain

Professor: Hongrui Liu

May 13, 2025

Table of Contents

[Chapter 1: Introduction 5](#_Toc197900869)

[1.1 Methodology 5](#_Toc197900870)

[Chapter 2: Data Collection 7](#_Toc197900871)

[Chapter 3: Network Diagram 10](#_Toc197900872)

[3. 1 Schedule and Demand Layer 10](#_Toc197900873)

[3.2 Travel Time Layer 11](#_Toc197900874)

[3.3 Space-Time Feasibility Layer 11](#_Toc197900875)

[Chapter 4: Problem Formulation 13](#_Toc197900876)

[Chapter 5: Results and Observation 19](#_Toc197900877)

[Chapter 6: Experimentation 21](#_Toc197900878)

[Chapter 7: Conclusion 25](#_Toc197900879)

[Chapter 8: References 27](#_Toc197900880)

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Description** | **Page No.** |
| 1 | Distance between SJSU and pick up locations in miles | 8 |
| 2 | Travel time between SJSU and pick up locations in minutes | 8 |
| 3 | Schedule and demand at pick up locations | 9 |
| 4 | Linearized Constraints Cases | 17 |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Description** | **Page No.** |
| 1 | Map of SJSU and pickup locations | 7 |
| 2 | Schedule and Demand of pickup locations | 10 |
| 3 | Travel time between locations in minutes | 11 |
| 4 | Space-time network diagram | 12 |
| 5 | Vehicle Path and Battery Levels when Bmax = 250 | 19 |
| 6 | Results from Python PuLP Optimization when Bmax = 250 | 20 |
| 7 | Vehicle Path and Battery Levels when Bmax = 200 | 21 |
| 8 | Results from Python PuLP Optimization when Bmax = 200 | 22 |
| 9 | Vehicle Path and Energy Tracking when Bmax = 350 | 24 |
| 10 | Results from Python PuLP Optimization when Bmax = 350 | 24 |
| 11 | Optimized EV Route Network | 25 |

# **Chapter 1: Introduction**

At the start of a new semester at San José State University (SJSU), the university's transportation department anticipates a significant influx of students, both international and domestic. To ensure smooth arrival and convenient commute to campus, the department is launching a student shuttle route service. This service will operate from major arrival points, including San Francisco International Airport (SFO) and San José International Airport (SJC). In addition, the department aims to support students already residing in campus dormitories by offering scheduled rides from essential destinations such as Costco Wholesale (San Jose), Westfield Valley Fair Mall (Santa Clara), and even Santa Cruz.

To promote environmental sustainability in line with the university’s “Go Green” initiative, electric vehicles (EVs) will be used instead of traditional gas-powered vehicles. To equitably serve all students, schedules and demand at each location are carefully estimated and incorporated into a capacitated routing model. The objective is to develop an efficient vehicle routing schedule that minimizes total travel time while meeting all demand and operational constraints.

## **Methodology**

To formulate the routing problem, travel times and distances between all locations are gathered using Google Maps. The SJSU campus serves as the primary depot, and the designated pickup locations include:

* San Francisco International Airport (SFO)
* San José International Airport (SJC)
* Westfield Valley Fair Mall (Santa Clara)
* Costco Wholesale (San Jose)
* Santa Cruz

The problem is modeled as a **Capacitated Electric Vehicle Routing Problem (CEVRP)**. Each electric vehicle has a defined battery capacity with a maximum limit and a minimum threshold. Energy consumption is modeled such that one unit of time equates to one unit of battery power. As the vehicle traverses between nodes, battery depletion is calculated, and if the battery level at any node falls below the minimum threshold, battery recharging is triggered. For simplicity, battery recharge is assumed to be instantaneous (i.e., time required for charging is not included in the model).

The complete mathematical formulation includes vehicle capacity constraints, battery level updates, node visit constraints, and route decision variables. This optimization problem is then implemented and solved using Python’s Pulp library, which allows for linear programming and integer programming modeling.

# **Chapter 2: Data Collection**

To design a viable and functional shuttle routing model, the team first gathered information about all relevant locations and constraints. The San José State University campus was designated as the depot from which all electric vehicles would begin and conclude their routes. The primary pickup points were San Francisco International Airport (SFO), San José Airport (SJC), San Jose Costco Wholesale, Westfield Valley Fair (WVF) in Santa Clara, and Santa Cruz city. Using Google Maps, the team estimated the distances from the depot to each of the individual pickup points, and the distances between the points. These distances were translated into travel times using a representative travel speed in which a mile is equated to a little over three minutes of road time, as a representation of typical traffic for shuttle operations.

**Figure 1.**

*Map of SJSU and pickup locations*

A map of a city

AI-generated content may be incorrect.

**Note.** *(0=SJSU, 1=SFO, 2=SJC, 3=Costco, 4=Santa Cruz, 5-WVF)*

**Table 1.**

*Distance between SJSU and pick up locations in miles*

A table with numbers and letters

AI-generated content may be incorrect.

Besides, demand and schedule information for each site were collected simultaneously. SFO and SJC airports, among others, were given two scheduled pickup times for several waves of incoming students. One scheduled pickup time based on access and volume anticipated was provided to the rest of the sites. Demand values, which were the numbers of students to be picked up at each stop, were estimated from historical transportation data, enrollment forecasts, and predicted residential distributions. These data were crucial in imposing vehicle capacity constraints on the optimization model.

**Table 2.**

*Travel time between SJSU and pick up locations in minutes*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **From/To** | **SJSU** | **SFO** | **SJC** | **COSTCO** | **SANTA CRUZ** | **WVF** |
| **SJSU** | 0 | 114 | 18 | 9 | 96 | 18 |
| **SFO** | - | 0 | 99 | 120 | 180 | 129 |
| **SJC** | - | - | 0 | 24 | 96 | 15 |
| **COSTCO** | - | - | - | 0 | 102 | 21 |
| **SANTA CRUZ** | - | - | - | - | 0 | 111 |
| **WVF** | - | - | - | - | - | 0 |

**Table 3.**

*Schedule and demand at pick up locations*

|  |  |  |
| --- | --- | --- |
| **Pick up locations** | **Schedule time** | **Demand** |
| **SFO - A** | 12:00 PM | 10 |
| **SFO - B** | 17:00 PM | 15 |
| **SJC - A** | 18:00 PM | 8 |
| **SJC - B** | 21:00 PM | 6 |
| **COSTCO** | 19:00 PM | 7 |
| **SANTA CRUZ** | 16:00 PM | 12 |
| **WVF** | 20:00 PM | 20 |

With the travel times, pickup windows, and corresponding demands, the team could then construct a full dataset for the problem. The data were employed as the foundation upon which to construct the routing network and set feasible arcs within the time-space network diagram. An arc would only be applicable when the vehicle could reach the next location within the available service time. The precision and completeness of the data ensured that the problem was correctly modeled, and optimal routing decisions considering spatial as well as temporal constraints were taken.

# **Chapter 3: Network Diagram**

To effectively represent the vehicle routing problem with time, capacity, and energy constraints, a multi-level network representation was created. The network not only captures the spatial distribution of service locations but also captures temporal and operational constraints critical to electric vehicle-based logistics. By progressively building the network in stages starting with schedule and demand, then travel times, and finally space-time feasibility. We have made sure that the final shape captures real-world constraints without being overly computationally intensive for optimization.

## **3. 1 Schedule and Demand Layer**

The start of the process of building was starting the identification of where and when the service is required. A varying scheduled time for service was assigned to each pickup point along with varying student demand. This was used to define the time windows in which each point is to be visited. SFO and SJC both had two different time windows (A and B), bringing in more routing flexibility but also more potential routes. This layer ensures that each node is not only covered but also served at the appropriate time with sufficient vehicle capacity.

**Figure 2.**

*Schedule and Demand of pickup locations*

A diagram of a schedule

AI-generated content may be incorrect.

## **3.2 Travel Time Layer**

On the second level, the travel time between all the nodes was estimated from Google Maps on the basis that one mile of travel distance equals approximately 3 minutes of travel time with average traffic conditions. This level provides spatial realism to the problem and is vital for modeling total cost (in terms of time) and energy consumption, as each minute of travel consumes one unit of battery from the electric vehicle. The travel time matrix is utilized as the foundation of arrival feasibility and battery usage between any two places.

**Figure 3.**

*Travel time between locations in minutes*

A diagram of a travel time

AI-generated content may be incorrect.

## **3.3 Space-Time Feasibility Layer**

The final and most significant step involved integration of the preceding two layers for the creation of a space-time network diagram. There were simply arcs (paths) in this layer that respected timing constraints retained. Specifically, an edge from node i to node j was included only if a vehicle could make the trip within the scheduled time: i.e., if Arrival Time ≤ Scheduled Time at j, where *Arrival Time* is calculated as Finish Time at i + Travel Time(i,j). This ensures that vehicles do not arrive at any pickup point late. The resulting network diagram captures all possible transitions, removes infeasible ones, and forms the basis for the optimization model to determine the most efficient, constraint-satisfying routes.

**Figure 4.**

*Space-time network diagram*

A diagram of a network

AI-generated content may be incorrect.

This layered process of network construction was essential to ensuring that the final model accurately captures both the physical and temporal feasibility of car travel. By removing infeasible arcs and retaining only those transitions consistent with demand timing and route reasoning, we laid a sound foundation for modeling the routing problem as a mixed-integer linear program. The resulting space-time network diagram not only illustrates the complexity of logistics but also allows for robust optimization that minimizes cost and meets schedule, capacity, and battery constraints.

# **Chapter 4: Problem Formulation**

When approaching the vehicle routing problem with real-world constraints like vehicle capacity and battery limits, we first translated the situation into a mathematical model that reflected how vehicles behave during their routes. This initial version naturally led to a non-linear formulation, since variables like load and battery level depended on whether a route was used resulting in equations that involved multiplying decision variables. While realistic, this kind of model is difficult for most optimization solvers to handle directly. To make it solvable, we used a process called linearization, which simplifies these complex expressions into linear ones without losing the logic behind them. We applied this to the capacity constraint, ensuring that no vehicle ever carries more than it should, and did the same for the battery constraint, which tracks how much charge is consumed during travel. Using the big-M method, we rewrote these rules in a way that optimization tools can process more efficiently. The result is a linear, solver-friendly model that still respects all the practical requirements helping us find optimal vehicle routes that meet demand, stay within capacity, and avoid running out of battery along the way.The objective of the model is to minimize the total time taken by vehicles to complete all required pickups. This is mathematically represented as below:

**Objective Function:**

A black and white symbol

AI-generated content may be incorrect.

**Constraints:**

1. **Flow Constraints:**

A math equation with numbers and symbols

AI-generated content may be incorrect.

A black and white math symbols

AI-generated content may be incorrect.

1. **Non-Linear Cumulative demand constraints (Capacity Constraints):**

A math equations and formulas

AI-generated content may be incorrect.

1. **Non-Linear Battery Flow constraints:**

A close-up of a math problem

AI-generated content may be incorrect.





1. **Non-Linear Recharge Trigger Constraints:**

A group of black text

AI-generated content may be incorrect.



1. **Binary Constraints:**

A math equations and formulas

AI-generated content may be incorrect.

A close up of a number

AI-generated content may be incorrect.

1. **Non-Negativity Constraints:**

All other variables (unless explicitly defined as binary) are assumed to be non-negative (i.e., ≥0)

**Definitions:**

*cij* is the travel time between node i and node j

*Xij* is a binary decision variable that equals 1 if a vehicle travels directly from node *i* to node *j*, and 0 otherwise.

*dj* is the demand at node j

*V* is the vehicle Capacity

*Vj* is the accumulated load at node j

*tij* is Travel time from node i to j

*rij* is 1 if recharging is triggered in arc(i,j), 0 otherwise

*bi* and *bj* are the battery level at node i and j

*Bmin* and *Bmax*are the Minimum and Maximum battery limits

N: is the node set, {0,1, 2....n}, with 0 depot

A:{(i,j) | i,j belongs to N, i not equal to j} is the Set of Arcs

The above formulation contains nonlinear constraints related to battery flow, recharging, and vehicle capacity. These constraints are linearized as follows:

1. **Vehicle Capacity constraints:**

We introduced the variable *Yij* to represent the product *Vi Xij,* which equals *Vi* if a vehicle travels directly from node i to node j (*Xij*=1), and 0 otherwise.

A black and white math equation

AI-generated content may be incorrect.

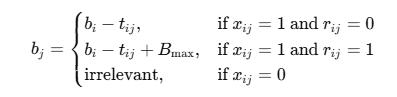
The following set of linear constraints ensures the correct behavior of *Yij* ​:

A black text with letters and numbers

AI-generated content may be incorrect.

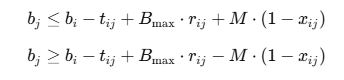
The above constraints ensure that if vehicle travels from node i to node j (i.e., *Xij=1* ), then *Yij = Vi;* otherwise, *Yij = 0.* Here, *W* is a sufficiently large constant such that *W >= Vi.* In this model we use the variable *W* in this constraint, instead of the traditional big-M symbol *M*, because it is used to represent *Bmax*, which is used for linearizing the battery constraints. The value for *W* is set equal to the maximum vehicle capacity (*V*), which is sufficiently large for this purpose.

1. **Battery flow constraints:**





The above equations are in nonlinear form. The battery level bj at node j depends on the battery level at the previous node i, the travel time tij, and whether the vehicle recharges at node i. When *Xij=0,* meaning the vehicle does not travel from node i to j, the right-hand side of the above equation becomes 0 and is therefore irrelevant. When the vehicle does travels from node i to j*, Xij*=1, and no recharging occurs at node i(*rij*=0), then the battery level at node j is simply the difference between the battery level at node i and the time required to traverse from node i to node j. Additionally, if the vehicle travels from node i to j, *Xij*=1, and recharges before departing (*rij*=1), then the battery level at node j is sum of the battery level at node i and the recharged amount *Bmax,* minus the travel time: *​bj=bi​−tij+Bmax​.* This expression is linearized using the big-M method as follows:



We introduce two inequalities (an upper and a lower bound) to bound the possible values of *bj* from both sides. These bounds ensure that the constraint behaves correctly when *Xij*=1, enforcing the battery update rule and avoiding partial recharging, while becoming non-binding when *Xij*=0. Additionally, battery levels are bounded at all nodes to ensure they remain within feasible limits:



The table below summarizes the behavior of the linearized equations under different values of *Xij*=0and *rij*.

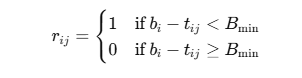
**Table 4.**

*Linearized Constraints Cases*

*A white background with black lines

AI-generated content may be incorrect.*

1. **Recharge Trigger Constraints:**



The recharge trigger equations are nonlinear in form. The underlying logic is as follows: when a vehicle travels from node i to j, *Xij*=1, the model first checks whether the available battery level at node i(*bi*)minus the travel time(*tij*)is sufficient. If this difference is less than the minimum battery threshold *(Bmin​),* the recharge indicator is activated by setting *rij*=1; otherwise*, rij =0*. If the arc (i,j) is not traversed(*Xij* =0), then *rij* =0. This constraint ensures that the vehicle recharges at node i before beginning its journey to node j, if needed. The linearized recharge trigger equations are as follows:



Here, *M* is a sufficiently large constant that relaxes the constraint when needed. When recharging is triggered (*rij* =1), the above equation forces the battery level at node 1 to be less than or equal to Bmin, which means recharging is necessary. When *rij* =0, the right-hand side becomes very large due to the term *M*(1− *rij*), and the constraint is relaxed - meaning no recharging required. When the battery level equation (*bi-tij*) is exactly equal to Bmin​, the model may arbitrarily choose whether to set rij​=1, due to numerical tolerances or solver behavior. To avoid this ambiguity, we introduce a small buffer (ε = 10-2), which ensures that even if the remaining battery is very slightly above the threshold, the model can safely avoid unnecessary recharging.



Additionally, the following constraint is used to enforce that the recharge indicator *rij* is only valid for selected routes, i.e., when *Xij = 1.*



# **Chapter 5: Results and Observation**

We implemented and solved the linearized model using Python with the PuLP optimization library in the Spyder environment. The model was provided with valid arc sets, travel times, and the demand at each pickup location as inputs. In the base configuration, the electric vehicle battery was defined with a maximum capacity of 250 units and a minimum operational threshold of 5 units. The maximum vehicle load capacity was set to 30 units.

From the results, the optimized objective value was found to be 687, and the model deployed 3 vehicles, each assigned to a distinct route, as illustrated in the figure below.

**Figure 5.**

*Vehicle Path and Battery Levels when Bmax = 250*

A diagram of a complex function

AI-generated content may be incorrect.

During the simulation, we observed that the recharge trigger variable activated correctly when required. Specifically, the recharge was triggered at node 1A. At the depot the vehicle is fully charged (b0=250). The vehicle travels from the depot (node 0) with a travel time of 114 units, resulting in a battery level of 136 upon arrival at node 1A (b1A = 250 − 114 = 136). The next leg of the journey was from node 1A to node 4, which required 180 units of travel time. Since the available battery at 1A was 136, which is less than the required 180, the model correctly triggered a recharge event, r[1A,4] = 1. Therefore, the battery level at node 4 after the recharge was calculated as:

b4 = Available battery(b1A) + Battery Max − Travel Time (t1A,4) = 136 + 250 – 180 = 206

This result confirms that the logic for triggering and applying recharges was functioning as intended. Across the entire solution, no unnecessary recharges were triggered elsewhere in the model. The total number of vehicles used was 3, and the overall cost was Z=687. Furthermore, the vehicle load capacity constraints at all visited nodes were valid and respected. The table below shows the values of the key decision variables as generated by the Python code after solving the model.

**Figure 6.**

*Results from Python PuLP Optimization when Bmax = 250*

A group of rectangular objects with numbers

AI-generated content may be incorrect.

# **Chapter 6: Experimentation**

In this section we tested how changing the battery capacity affects the overall optimization. As expected, altering the battery capacity led to changes in the objective function value, since different recharge strategies, route feasibility, and energy limitations influenced the total travel time. This demonstrates that the battery constraint is not only active but also meaningfully impacts the routing decisions and optimization results.

To validate the effectiveness of the formulated model, we conducted experiments with 200 and 300 units of maximum battery capacity. These experiments revealed that the battery constraint meaningfully impacts the optimization outcome. At 200 units, the system required 4 vehicles and showed longer total travel time with an overall cost of Z=735 due to limited routing flexibility and more frequent recharges.

**Figure 7.**

*Vehicle Path and Battery Levels when Bmax = 200*

A diagram of a mathematical equation

AI-generated content may be incorrect.

**Figure 8.**

*Results from Python PuLP Optimization when Bmax = 200*

A graph of numbers and letters

AI-generated content may be incorrect.

On the other hand, increasing the battery capacity to 300 units gave more freedom but didn’t result in a significantly better outcome it sometimes even led to suboptimal routing because the relaxed energy constraint reduced pressure on efficient planning. In this case, the model used 4 vehicles without requiring any recharging, with an overall cost of Z=702.

**Figure 9.**

*Vehicle Path and Energy Tracking when Bmax = 350*

A diagram of a mathematical model

AI-generated content may be incorrect.

**Figure 10.**

*Results from Python PuLP Optimization when Bmax = 350*

A group of numbers on a white background

AI-generated content may be incorrect.

The best outcome across all trials was observed at 250 units, which yielded the lowest total travel time and required just three vehicles. This balance between energy availability and route feasibility made it the most efficient configuration. Overall, the model demonstrates sensitivity to battery capacity and shows how well-designed recharge logic and route planning can significantly influence the time efficiency of electric vehicle operations.

***Figure 11.***

*Optimized EV Route Network*

A diagram of a network

AI-generated content may be incorrect.

# **Chapter 7: Conclusion**

The Capacitated Electric Vehicle Routing Problem (CEVRP) developed in this project successfully addresses the challenge of optimizing student shuttle operations at San José State University, while incorporating realistic constraints such as vehicle load limits, battery capacity, and time-based service schedules. A structured, multi-layered network was constructed starting from schedule and demand data, followed by travel time estimation, and culminating in a space-time feasibility network. This layered approach ensured that only time- and energy-feasible arcs were considered for optimization.

To manage the complexity of the problem, which included non-linear battery and capacity constraints, a robust linearization strategy using the big-M method was applied. This enabled the modeling of conditional battery flow and recharge logic within a Mixed Integer Linear Programming (MILP) framework using Python’s PuLP library. The space-time network diagram and battery status tracking validated the accuracy of vehicle assignments and recharge triggers when energy thresholds were reached. At node 1A, for instance, the vehicle’s remaining battery was insufficient for the next segment, and the recharge was correctly activated, demonstrating the model's precise control over battery constraints.

Extensive experimentation showed that battery capacity significantly impacts routing efficiency. A lower capacity (200 units) resulted in more frequent recharges, longer travel time, and the use of four vehicles. In contrast, a higher capacity (300 units) eliminated the need for recharging but sometimes led to suboptimal routing due to over-flexibility. The optimal outcome was achieved at 250 units, where only three vehicles were required, and the total travel time was minimized, representing the most balanced and efficient solution.

In conclusion, the model provides an effective and sustainable approach to EV-based student transportation and highlights the importance of integrating energy constraints into route planning. The framework lays a strong foundation for future improvements, such as incorporating fixed recharging station locations, accounting for charging durations, or enabling real-time route adjustments to enhance practicality and scalability.

**7.1 Future Scope**

The current model effectively addresses vehicle routing under battery and capacity constraints; however, several opportunities exist to enhance its realism and applicability. One key improvement would be incorporating the cost of recharging into the objective function. At present, energy usage is not factored into operational expenses, but in real-world applications, this is a significant consideration. Integrating energy consumption or recharging costs would yield a more accurate representation of total operating costs. Additionally, the current model assumes instantaneous recharging, which simplifies calculations but neglects the time required to recharge vehicles, an important factor in practical scheduling. Future models should incorporate charging time as a constraint, accounting for the delays it introduces and how it affects route feasibility. Another limitation is the assumption of universal recharge availability at all nodes, which is unrealistic. A more practical approach would be to restrict recharging to specific, predefined locations. Defining dedicated recharging stations within the model and ensuring that routes naturally incorporate them when needed would better align with existing electric vehicle infrastructure. Furthermore, the inclusion of multiple recharging stations and the development of a location strategy possibly using nearest neighbor heuristics, k-means clustering, or hybrid optimization techniques such as mixed-integer linear programming combined with local search could significantly improve the model’s scalability and realism in large-scale applications.

# **Chapter 8: References**

1. Erdelic, T., Carić, T., Erdelic, M., & Tišljarić, L. (2019). Electric vehicle routing problem with single or multiple recharges. *Transportation Research Procedia, 40*, 217–224.

<https://doi.org/10.1016/j.trpro.2019.07.033>

1. Conrad, R. G., & Figliozzi, M. A. (2011). The recharging vehicle routing problem. In T. Doolen & E. Van Aken (Eds.), *Proceedings of the 2011 Industrial Engineering Research Conference* (pp. 18). <https://api.semanticscholar.org/CorpusID:110272876>
2. Hashimoto, H., Hu, Y., & Okamoto, Y. (2022). An efficient route evaluation method for the vehicle routing problem with linear constraints. In 2022 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) (pp. 511–515). <http://alg.u.e.kaiyodai.ac.jp/publication_en.html>