CIRRELT
1971 – 2021

CIRRELT-2021-25

Static Bike Relocation Optimization Applied to Montreal's BIXI Bike Sharing System

Imad Mechmachi
Philippe Lavoie
Reda Kzaz
Jean-Yves Potvin

July 2021

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

Static Bike Relocation Optimization Applied to Montreal's BIXI Bike Sharing System[ł]

Imad Mechmachi[1], Philippe Lavoie[1], Reda Kzaz[1], Jean-Yves Potvin[1,2]

[1] Department of Computer Science and Operations Research, Université de Montréal

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

Abstract. This paper presents a variable neighborhood search heuristic for the static bike relocation problem applied to Montreal's BIXI bike sharing system. We also present an original algorithm for estimating optimal relocation quantities. Focus is on night relocation operations, which is considered to be a static problem since bike flow is negligible. We first present a demand and bike transition model for user behavior and utilize it to estimate optimal relocation quantities by means of stochastic simulations. We then solve a single visit routing problem using a fleet of vehicles for optimal relocation operations routes. Finally, solution visualizations and statistics are given for the bike-sharing system in Montreal.

Keywords: Bike sharing system, bike relocation, demand modeling, vehicle routing, variable neighborhood search.

ł This paper has won the First Prize of the Student Paper Competition, under-graduate category, of the Canadian Operational Research Society (CORS).

---

* Corresponding author: potvin@iro.umontreal.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2021

1 Introduction

Bike sharing systems are a collection of stations (or docks) and bikes that users can rent out to make trips from one station to another. Better bike availability, through relocation operations, is crucial and has been showed to correlate directly with increases in ridership [6]. A bike sharing system usually employs a team of logistics planners and relocation drivers to manage and relocate bikes all around the clock. Relocation in large networks is a challenging problem. During the day, bikes are flowing and relocation planning is dynamic. Thus, strong predictive models for demand and fast vehicle routing optimization are required. However, a significant portion of rebalancing operations occurs at night where bike flow is negligible. In this context, we refer to the static bike relocation problem. This will be the focus of our study and the application for Montreal's bike sharing system.

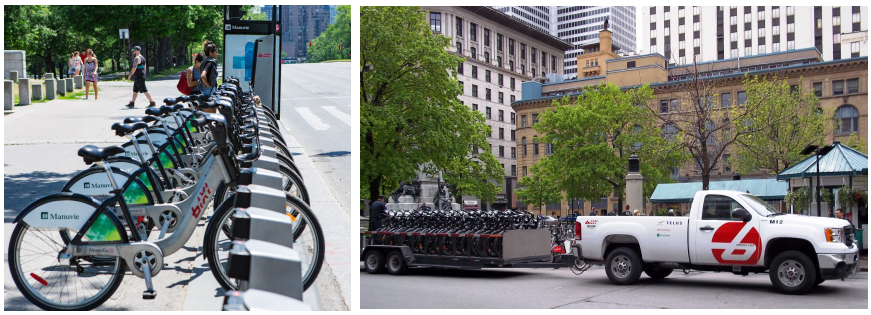


Figure 1: A dock station is on the left and a relocation truck on the right. Both from Montreal's BIXI bike sharing system. (Sources : left, right)

Montreal's BIXI bike-sharing system which began operations in 2009 figures among the largest networks in North America. It has more than 600 stations (highly varying number) with well over 7400 bikes and has provided, at its peak, more than 5.7 million trips during 2019 alone. Network operations usually span only from April to October because of the notorious cold weather in Montreal. The network is also remarkably dense with more than 90% of stations having a neighbor less than 1 km away and around 70% less than 500 m away, as shown in Figure 2. In this paper, we worked on data coming from summer 2019 for availability reasons and pre-COVID-19 settings (a sharp -60 % drop in the number of trips was recorded in 2020). We considered 619 of the most long-lasting stations in 2019 (since some stations are only temporarily implemented).

We also address single-visit routing with fixed relocation quantities. Thus, we estimate the quantities of bikes to pickup or deliver for each station first and solve the vehicle routing problem second.

A lot of work has already been done in bike-sharing system optimization. C.S. Shuia and W.Y. Szeto [11] provide an extensive literature review on the subject and classify the many problems related to bike relocation. The static bike relocation problem (SBRS) is known to be NP-hard (Benchemol et al. [7]), which justifies the need for

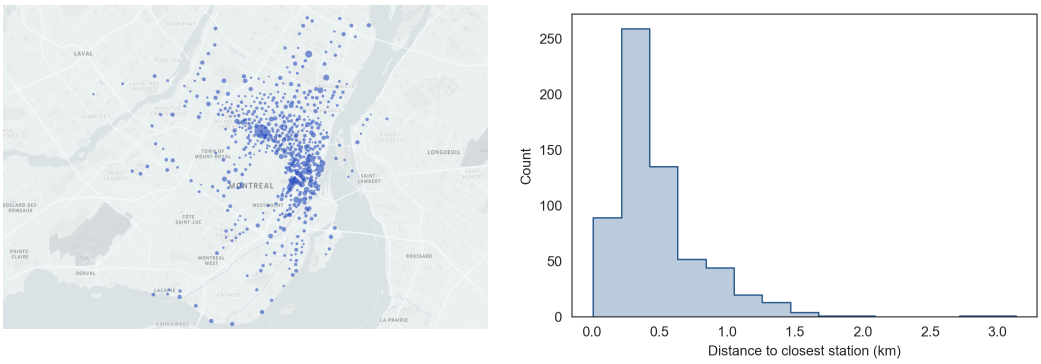


Figure 2: On the left : station locations in Montreal with radius reflecting capacity of each station. On the right : closest station route distance distribution in Montreal, noting the high density of the network

heuristic algorithms. Our data structures and adjusted cost function are mostly inspired by Teobaldo Bulhões et al. [1]. In this paper, the authors successfully solve the static problem with multiple visits and provide substantial computational experiments (note that we focus on single visits in our work). Many other successful metaheuristic implementations are reported in the literature, like Yaping Ren et al. [9] or Sin C. Ho and W.Y. Szeto [5] who also provide extensive computational experiments.

In this paper, we first propose a mathematical formulation and demand model in Section 2. Next, we introduce an algorithm for relocation quantity estimation in Section 3. The heuristic algorithm for vehicle routing is then presented in Section 4. Solutions for Montreal's BIXI system are reported in Section 5. Finally, in Section 6, we conclude with some remarks.

It should be noted that an important contribution of this paper is the novel algorithm described in 3.4 for estimating pick-up or delivery quantities through simulations.

2 Model

Here we give the full mixed integer linear program (MILP) for the vehicle routing problem, which is inspired by Teobaldo Bulhões et al. [1]. This is followed by a predictive demand model for BIXI's network.

2.1 Mathematical Formulation

We suppose that bike pick-up or delivery quantities q_i for every station i are given ($q_i > 0$ indicates a pick-up and $q_i < 0$ a delivery). We represent the network of stations with a directed graph $G = (V, A)$ by only considering stations to be affected by bike relocation (that is $q_i \neq 0$). Let 0 and f represent the starting and final node, respectively, both at the same depot location. The graph with nodes $V \setminus \{0, f\}$ is

complete, whereas 0 has leaving arcs to all stations and f has entering arcs from all stations. We define $A^-_{(i)}$ and $A^+_{(i)}$ the set of entering and leaving arcs, respectively, for node i.

Let N be the number of available trucks, C the truck capacity, T the time limit for each truck, c_{ij} the cost from station i to j (in practice, we use a combination of time and route distance $c_{ij} = \alpha t_{ij} + \beta d_{ij}$, which is considered to be time-independent due to low traffic rates during the night). We also use a fixed handling time per bike δ and per station visit δ_0 such that the total time elapsed for the handling of a station is $\delta_0 + \delta|q_i|$.

Decision variables :

- x_{ij} has value 1 if arc (i, j) is taken and 0 otherwise.
- u_{ij} bike flow on arc (i, j).
- Δ_{ij} elapsed time after traversing arc (i, j) (include handling time for i but not for j).

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

subject to

$$\sum_{(0,j)\in A^+_{(0)}} x_{0j} \leq N \tag{2}$$

$$\sum_{(i,j)\in A^+_{(i)}} x_{ij} - \sum_{(j,i)\in A^-_{(i)}} x_{ji} = 0 \qquad i \in V \setminus \{0, f\} \tag{3}$$

$$\sum_{(0,j)\in A^+_{(0)}} x_{0j} - \sum_{(j,f)\in A^-_{(f)}} x_{jf} = 0 \tag{4}$$

$$\sum_{(i,j)\in A^+_{(i)}} x_{ij} \leq 1 \qquad i \in V \setminus \{0, f\} \tag{5}$$

$$\sum_{i,j\in S, i\neq j} x_{ij} \leq |S| - 1 \qquad \{0, f\} \not\subset S,\ S \subset V \tag{6}$$

$$\sum_{(i,j)\in A^+_{(i)}} u_{ij} - \sum_{(j,i)\in A^-_{(i)}} u_{ji} = q_i \qquad i \in V \tag{7}$$

$$u_{ij} \leq Cx_{ij} \qquad (i, j) \in A \tag{8}$$

$$\sum_{(i,j)\in A^+_{(i)}} \Delta_{ij} - \sum_{(j,i)\in A^-_{(i)}} \Delta_{ji} = \sum_{(i,j)\in A^+_{(i)}} t_{ij}x_{ij} + \delta_0 + \delta|q_i| \qquad i \in V' \setminus \{0, f\} \tag{9}$$

$$\Delta_{0j} = t_{0j}x_{0j} + \delta_0 + \delta u_{0j} \qquad (0, j) \in A^{(+)}_{(0)} \tag{10}$$

$$\Delta_{jf} + \delta_0 + \delta u_{jf} \leq T x_{jf} \qquad (j,f) \in A^{(-)}_{(f)} \tag{11}$$

$$\Delta_{ij} \leq T x_{ij} \qquad (i,j) \in A \tag{12}$$

$$x_{ij} \in \{0,1\} \quad u_{ij} \in \mathbb{Z}_{\geq 0} \quad \Delta_{ij} \geq 0 \tag{13}$$

Equation (1) is for travel costs minimization, (2) limits the number of trucks leaving the depot to N, (3), (4), (5) are for path connections and (6) bans subtours that do not cover depot nodes 0 and f. Flow conservation is given by (7), whereas (8) limits bike flow to truck capacity. The time update is given by (9) (with t_{ij} the route travel time from i to j), (10) initializes time and (11), (12) bound total operations time for each truck by T.

2.2 Demand and Transition Model

We describe here the stochastic model for user arrivals prediction and bike transitions in the network, as illustrated in Figure 3. First, let us define :

- $I_i(t)$ number of bikes at station i at time t (inventory).
- $r_i(t)$ number of user arrivals from the beginning of the day to time t at station i.
- L_i bike limit (maximum number of bikes that can be held in station i : $I_i \leq L_i$).

User arrivals $r_i(t)$ are people interested in taking a bike at station i and either took one or did not find one because the station was empty (and may be took one from a neighboring station). We model user arrivals with a non-homogeneous Poisson process :

$$r_i(t) \sim \text{Poisson}\left(\int_0^t \lambda_i(z, data)dz\right)$$

The rate arrival $\lambda_i(t, data)$ depends on the time of the day t and relevant $data$ such as calendar date and meteorological data.

When a user arrives at a non-empty station i, it translates into a bike departure at i. However, if the station is empty, it either translates into a bike departure at a neighboring station $neighborhood(i)$ if reasonable walking distance permits or nothing.

Once a departure is set, the destination is modeled using transition probability matrix $p_{ij}(t, data)$, which denotes the probability of performing a transition to station j given the origin station i :

$$destination(i) = j \sim \text{Discrete}(p_{ij}(t, data)) \qquad p_{ij}(t, data) = \mathbb{P}[j \mid i, t, data]$$

The bike travel time given the origin and destination is modeled using a normal distribution with mean μ_{ij} and standard deviation σ_{ij}. Furthermore, if a bike is planned to arrive at a full station, it is redirected to a neighboring station. We simply used the closest available station for $neighborhood(i)$.

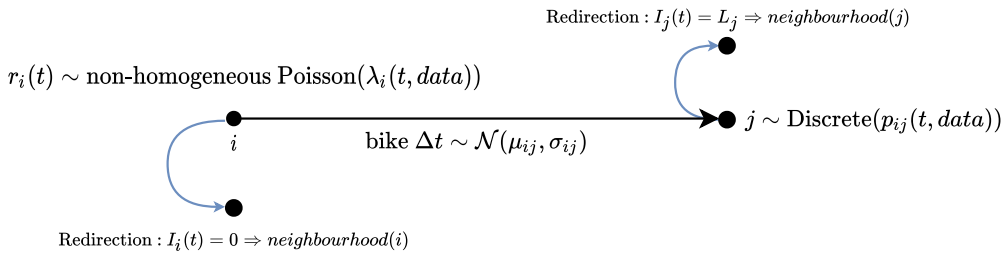


Figure 3: Model diagram. This diagram is also relevant for simulation details in 3.3

3 Estimation

In this section, we go over techniques that are used to estimate parameters for the transition and demand model described in 2.2. We also explain how we estimate the pick-up and delivery quantities.

3.1 Data

Montreal's BIXI organization, thanks to its open data policy, provides detailed information and historical data on the network. We have access to a complete list of all bike trips for every year from 2014 to 2020 (see Open Data BIXI). Data points are formatted (start date, start station code, end date, end station code). In 2019, there were more than 5.6 million entries. Each entry corresponds to a specific trip. Here is a sample :

start date	start station code	end date	end station code
2019-04-14 07:55:22	6001	2019-04-14 08:07:16	6132
2019-04-14 07:59:31	6411	2019-04-14 08:09:18	6411
2019-04-14 07:59:55	6097	2019-04-14 08:12:11	6036
...	...	...	...

We also have access to additional data about the BIXI's network including the number of members, station locations, current station inventory and bike limits for each station (using the GBFS API).

Historical weather and climate data are provided by Environment Canada weather historical data. We also have all route distances and route travel times between stations, as given by Bing Map distance matrix API. We worked on data from 2019 for reasons given in Section 1.

3.2 Parameter Estimations

We divided the data into 3 sets: training (93.3 %), validation (20% cross validation of training set) and test (6.7%) sets. We estimate the user arrival rate $\lambda(t, data)$ by fitting a regression to the observed average rate (# arrivals in the interval / interval length) with gradient boosting decision trees ensemble model [2]. We used 1h intervals

and the following carefully crafted features : Time, Month, Weekday?, Temp (°C), Climate, Wind speed (km/h), rel. Hum. (%). For hyper-parameter selection, we plot validation curves for the number of estimators (decision trees) and the maximum decision tree depth, see Figure 4. It should be noted that the low R^2 scores come from the very sparse data obtained when we take off 20% of the training set to plot the validation curves. When the model is trained on the whole training set, much better results are obtained.

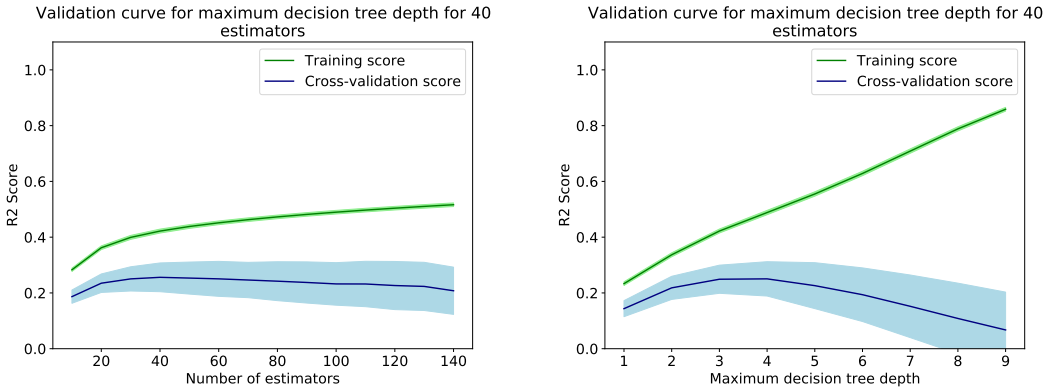


Figure 4: Validation curves R^2 for two hyper-parameters. 95% confidence intervals are shown in blue.

To avoid over-fitting and under-fitting, we select parameters that maximize the cross-validation score. For the final model, the number of estimators and the maximum decision tree depth are set to 40 and 4 respectively [8]. The final model got a R^2 score of 0.25 with a standard deviation of 0.06 for a 5 fold cross-validation. The R^2 score for the test set is 0.397, which we find to be good enough considering the randomness of the phenomena and the high number of predicted variables (one for every 619 stations for 24h).

The graphics of cumulative arrivals, depending on the time of the day, are plotted in Figure 5 to see the performance of the model on the test set. The prediction's 95% confidence interval is shown in light-blue. Confidence intervals are estimated using a normal distribution.

Figure 6 shows the number of arrivals for all stations depending on the time of the day. We split the network into two categories of behavior: commute and residential. Note that commute stations are more likely to have arrivals after 15h whereas stations of the residential type are more likely to have arrivals before 10h. We can see that model predictions are close to observed test data. We also notice that stations near the commute destination have a lot more arrivals after 15h and stations closer to residential areas have more arrivals before 10h : this behavior has been captured by the model.

Finally, to complete the estimation of parameters, transition probabilities $p_{ij}(t, data)$ are estimated using average observed origin-destination matrices given by the data

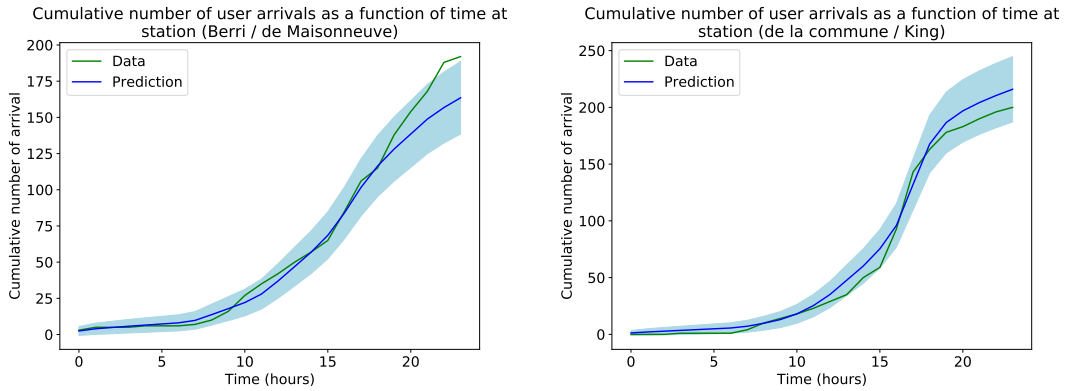


Figure 5: Cumulative number of user arrivals predictions on test data, July 12, 2019, a partly cloudy weekday

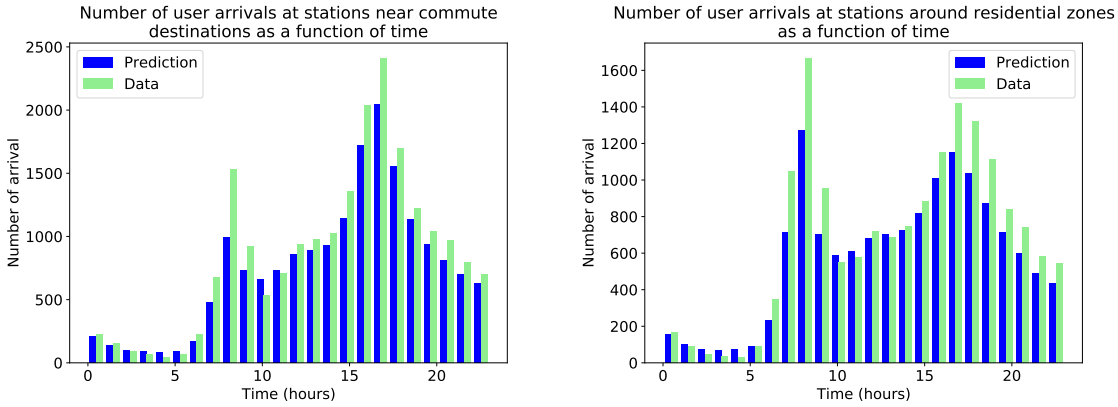


Figure 6: Total user arrivals predictions on test data, July 12, 2019, a partly cloudy Friday

set. Each 3h block during the day is assigned a transition matrix and we use a different set of matrices during weekends. For trip duration, we estimate μ_{ij} using the average duration for trip from i to j in the dataset and we use 25% standard deviation σ_{ij}.

3.3 Simulations

Given a current network inventory $I(t)$ (current number of bikes $I_i(t)$ in each station) we can perform simulations given by our transition and demand models to study the dynamics of $I_i(t)$.

The simulation is straightforward: we sample user arrivals from our non-homogeneous Poisson process using standard methods presented by Ross [10]. We assign a destination and travel time to each user arrival with our transition probability matrix p_{ij}

that corresponds to the correct time period and we sample travel duration from a normal distribution. We also redirect user arrivals at empty stations or bike arrivals at full stations to the closest station from the target. For more insights, refer to Figure 3.

During simulation, we count the number of missed user arrivals $M^{(a)}$ and missed bike returns $M^{(r)}$ (that is, a user arriving in an empty station and a bike returning to a full station, respectively). Those quantities are crucial to the estimation of pick-up and delivery quantities, since we want to minimize the number of misses. Let $simulation()$ be the function that returns an estimation of the expected values of $M^{(a)}$ and $M^{(r)}$ by computing some sample mean through multiple simulations, with t_0, t_1, $data$ being the start time, end time, data features, respectively, and $I = I(t_0)$ being the network inventory at time t_0, as defined in 2.2.

$$(\hat{M^{(a)}}, \hat{M^{(r)}}) = \text{simulation}(I, t_0, t_1, data)$$

3.4 Pick-up and Delivery Quantities Estimation

The ultimate goal of developing the statistical model is to harness its predictions to compute a good estimation of pick-up and delivery quantities at each station for relocation operations. We present here a simple heuristic to obtain such an estimate using the simulation outputs.

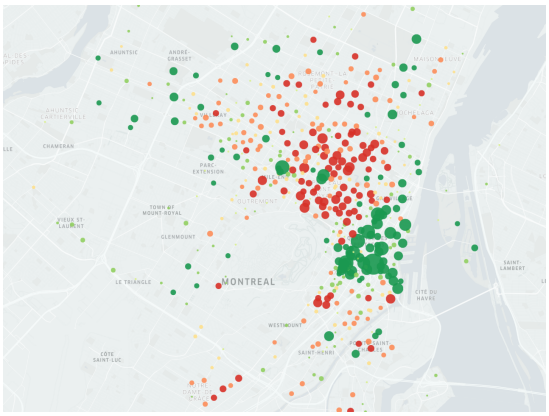
Knowing that we are working on night operations, let us focus on minimization of the expected values of the number of missed arrivals $M^{(a)} + M^{(r)}$ during next day morning, say $t_0 = 5\text{h}$ to $t_1 = 12\text{h}$, given a modified initial inventory state vector $I(t_0) - q$:

$$(q_i^*)_{i \in V} = \underset{q_i | i \in V}{\arg\min} \sum_{i \in V} \mathbb{E}\left[M_i^{(a)} + M_i^{(r)} \,\middle|\, I(t_0) - q \right]$$

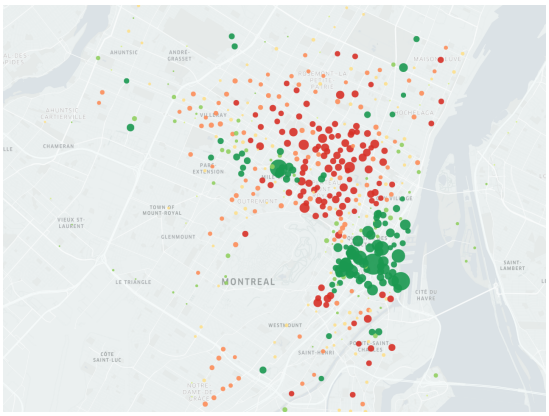
where the q_i values are the pick-up and delivery quantities described in 2.1 and $q = (q_i)_{i \in V}$. An interesting alternative is to weight the expectation for station i according to some discomfort metric, for example the distance to the closest station. The key idea is to estimate the expectation using simulation outputs :

$$\mathbb{E}\left[M_i^{(a)} + M_i^{(r)} \,\middle|\, I(t_0) - q \right] \approx \hat{M_i^{(a)}} + \hat{M_i^{(r)}}$$

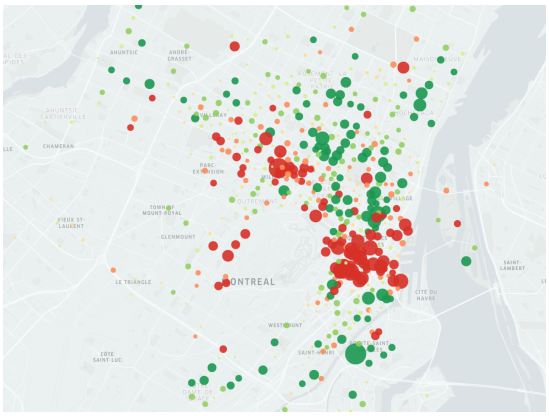
A simple heuristic to estimate good values for q_i is simply to increment inventory I_i in the direction of $\hat{M_i^{(a)}} - \hat{M_i^{(r)}}$ and get $q_i = -\Delta I_i$ where ΔI_i is the change in station inventory obtained by the relocation operations. The intuition is that to reduce large values of missed user arrivals at empty stations $\hat{M_i^{(a)}}$, we can augment inventory levels I_i and for large values of missed bike returns at full stations, we can decrease inventory levels. Algorithm 1 below is is a simple heuristic to capture this idea.



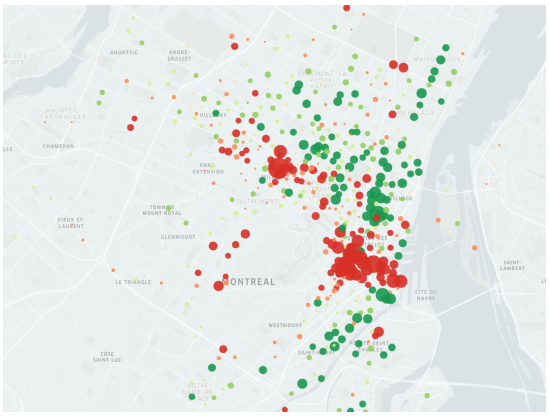
(a) Morning predictions



(b) Morning test data



(c) Late afternoon/evening predictions



(d) Late afternoon/evening test data

Figure 7: Bike flux (number of bike arrivals - number of bike departures) for all stations. Positive flux is in green (meaning hot destination) and negative flux is in red (meaning important origin or leaving point). Radius size around each station reflects flux magnitude. From test data, July 12, 2019, a Friday.

Note that in the morning, bikes are leaving residential-like areas in direction of Downtown Montreal and in late afternoon this is the opposite. This shows a typical commute to work weekday.

Algorithm 1 Heuristic for estimation of relocation quantities $\hat{q}_i \approx q_i^*$

Input : Initial inventory condition $I(t_0)$, optimization period end day time t_1
Parameters : Number of iterations n, $\theta_k > 0$ constants
Output : q_i estimates

$S_i \leftarrow I_i(t_0), \qquad i \in V$
$\Delta S_i \leftarrow 0, \qquad i \in V$
$R \leftarrow 0$
for $k = 1, ..., n$ **do**
  $\hat{M_i^{(a)}}, \hat{M_i^{(r)}} \leftarrow \text{simulation}(S, t_0, t_1, data)$
  $S_i \leftarrow S_i + \theta_k(\hat{M_i^{(a)}} - \hat{M_i^{(r)}}) - R \cdot |\Delta S_i| \qquad i \in V$
  $S_i \leftarrow \text{Clip}_{0,L_i}(S_i)$
  $\Delta S_i \leftarrow S_i - I_i(t_0) \qquad i \in V$
  $R \leftarrow \frac{\sum_{i \in V} \Delta S_i}{\sum_{i \in V} |\Delta S_i|}$
end
$\hat{q}_i \leftarrow -\Delta S_i \qquad i \in V$
Return $\hat{q}_i$

In this heuristic, $\text{Clip}_{0,L_i}(S_i)$ clips S_i to the bounds of interval $[0, L_i]$ when it is outside of that interval (L_i is the bike station limit). The $-R \cdot |\Delta S_i|$ term in the update insures rebalancing only, that is $\sum_{i \in V} \Delta S_i \approx 0$ (minimal network bike variation to be handled at the depot). The aim is to reduce bike excess or deficit $\sum_{i \in V} \Delta S_i$ by redistributing it proportionally to $|\Delta S_i|$. This can be easily seen by rewriting $-R \cdot |\Delta S_i| = -(\sum_{i \in V} \Delta S_i) \cdot |\Delta S_i| / \sum_{i \in V} |\Delta S_i|$ with $|\Delta S_i| / \sum_{i \in V} |\Delta S_i|$ the proportional factor. In our experiments, we optimized for period $t_0 = 5$h to $t_1 = 12$h and ran the algorithm for $n = 1$ to 20 iterations. We also used a linearly decreasing $\theta_k = 0.8 \cdot (1 - (k-1)/n)$.

As shown in Figures 8 and 9, Algorithm 1 is successful for estimating good q_i relocation quantities and keeps improving its estimate even after multiple iterations ($n > 4$). However, performance seems to hit a plateau after around $n = 10$. Also, we visualize only target arrivals or return attempts (not redirection).

4 Vehicle Routing Heuristic

In this section, we overview the heuristic for finding solutions to the vehicle routing problem described in 2.1.

4.1 Variable neighborhood Search

After building an initial solution (which is explained in 4.4), we improve it with the variable neighborhood search metaheuristic, as presented by P. Hansen et al. (2010) [4]. However, instead of shaking a solution, we perform a number of tabu

(a) Without relocation

(b) $n = 1$

(c) $n = 3$

(d) $n = 8$

Figure 8: Number of missed user arrivals at empty station $\hat{M^{(a)}}$ in purple and number of missed bike returns at full station $\hat{M^{(r)}}$ in red (estimated by simulation) as a function of relocation strategy given by n iterations of Algorithm 1. Radius size around each station reflects the magnitude of the number of missed arrivals. From test data, 5h00 to 12h00, June 1, 2019, a Saturday. Note that without relocation, missed bike returns at full stations (in red) is more common in the Downtown area and the reverse is true for residential-like areas (purple). This is to be expected and it reflects the standard commute pattern in Montreal.

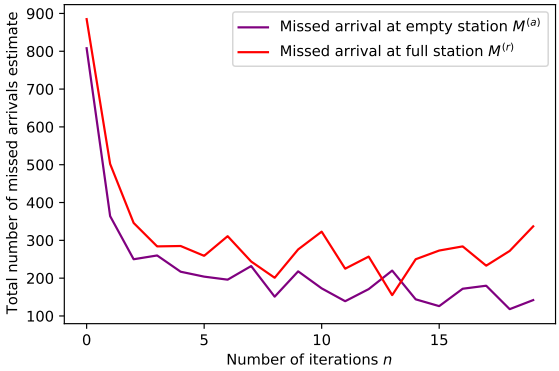


Figure 9: Total missed arrivals estimate (by simulation) as a function of relocation strategy given by n iterations of Algorithm 1. From test data, 5h00 to 12h00, June 1, 2019, a Saturday.

search iterations on a randomly chosen neighborhood [3]. This step is then followed by a variable neighborhood descent (VND) and the process is repeated until the computation time limit t_{max} is reached. Futhermore, we allow infeasible solutions during the search by using an adjusted cost function described in 4.3.

During the tabu search iterations, we temporarily reduce the penalty weights associated with capacity constraints and, at each iteration, a neighborhood from 2-opt, reinsert, 2-opt* is selected and the best solution found is returned. Three consecutive tabu search iterations are performed before going back to the VND. In the latter, the ordering of the neighborhoods is fixed and a solution is replaced as soon as an improving solution is encountered. Here is the pseudo-code :

Algorithm 2 Variable neighborhood search scheme.

$x \leftarrow$ build initial solution(Network, q_i)
while $t < t_{max}$ **do**
  $x' \leftarrow$ Tabu Search iterations(x)
  $x' \leftarrow$ VND(x')
  $x \leftarrow$ Best(x, x')
end
Return x

In the pseudo-code, Best(x, x') returns the best solution between x and x' according to the adjusted cost Z given in 4.3. VND(x) corresponds to the variable neighborhood descent algorithm [4]. As mentioned before, we perform three tabu search iterations and we choose each time a random neighborhood from 2-opt, reinsert and 2-opt*. We also keep the tabu list during VND to forbid reversing changes done by the tabu steps.

4.2 Definitions and Auxiliary Data Structures

We present here the auxiliary data structures used for implementing the heuristic, as given by Bulhões et al. (2018) [1]. Each route performed by truck $0 \leq k \leq N$ is denoted by an ordered sequence $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_{m_k-1})$ of stations, where m_k denotes the number of station visits by truck k. The notation $\sigma_{i,j}$ represents a sub-route of σ that begins at index position i and ends at index position j. We define the following formulas which will be useful for the evaluation function.

- $q_{sum}(\sigma) = \sum_{i=0}^{m_k-1} q_{\sigma_i}$, sum of picked-up/delivered bikes on route σ (cumulative load).
- $q_{min}(\sigma) = \min(0, q_{sum}(\sigma_{0,0}), q_{sum}(\sigma_{0,1}), \ldots, q_{sum}(\sigma_{0,m_k-1}))$ minimum cumulative load.
- $q_{max}(\sigma) = \max(0, q_{sum}(\sigma_{0,0}), q_{sum}(\sigma_{0,1}), \ldots, q_{sum}(\sigma_{0,m_k-1}))$ maximum cumulative load

We store for each i, j, with $i \leq j$, the values of $q_{sum}(\sigma_{ij}), q_{min}(\sigma_{ij})$ and $q_{max}(\sigma_{ij})$. We also store in memory 1-d vectors that contain cumulative costs and time values (and their reversed counterparts for 2-opt cost adjustment computations; note that we do not assume symmetric arc costs). That is, we have $c(\sigma)$ $c(\overleftarrow{\sigma})$, $t_{tot}(\sigma)$ and $t_{tot}(\overleftarrow{\sigma})$ for every sub-route σ_{0i} stored :

- $c(\sigma) = \sum_{i=0}^{m_k-2} c_{i,i+1}$, gives the cumulative cost of route σ.
- $t_{tot}(\sigma) = \sum_{i=0}^{m_k-2} t_{i,i+1} + \sum_{i=0}^{m_k-1}(\delta_0 + \delta|q_i|)$, gives the total time of route σ.

Having data on every sub-route stored, cost evaluation and feasibility test of neighbor solutions (2-opt, shift, swap, etc.) can be efficiently computed in $O(1)$ time.

4.3 Adjusted Cost Function

During local search, we adjust the cost function to allow infeasible solutions through penalty weights γ_Q and γ_T [1]. Note that x^+ denote the ReLU function $x^+ = \max(0, x)$ and $x^- = \min(0, x)$:

$$Z(\sigma) = c(\sigma) - \gamma_Q(q_{min}(\sigma))^- + \gamma_Q(q_{max}(\sigma) - C)^+ + \gamma_T(t_{tot}(\sigma) - T)^+ \qquad (14)$$

In this cost function, $c(\sigma)$ is the objective travel costs, $-\gamma_Q(q_{min}(\sigma))^-$ and $\gamma_Q(q_{max}(\sigma) - C)^+$ are penalties for not satisfying truck capacity constraints (that is when $q_{min} < 0$ or $q_{max} > C$) and $\gamma_T(t_{tot}(\sigma) - T)^+$ is for not satisfying maximum total time T. Weights γ_Q and γ_T are fixed except during the tabu search steps where they are temporarily reduced.

4.4 Construction of Initial Solution

Initial solutions are built using a simple greedy insertion heuristic. That is, we first initialize N routes with the depot nodes and we consider the best possible combination of insertion position $i*$, node $s*$ and truck $k*$. Once the insertion is performed, we repeat until all nodes are done.

$$k*, i*, s* = \arg\min_{k,i,s} Z(\sigma_{0,i} \oplus s \oplus \sigma_{i+1,|\sigma^{(k)}|-1})$$

We have $k \in \{1, ..., N\}$, $i \in \{0, ..., |\sigma^{(k)}| - 2\}$ and $s \in V \setminus \sigma^{(k)}$ during the search. Evaluation of Z can be done in $O(1)$ by using the auxiliary data structures and update formulas given in A.6.

4.5 Neighborhoods

As it is common practice, we divide our neighborhoods into two classes : intra-route and inter-route neighborhoods.

We use the following inter-route neighborhoods :

- Shift : transfer a station from one route to another.
- 2-opt* : Two routes are cut. Then, the first part of route 1 is reconnected with the second part of route 2 and vice versa (see Figure 10). The distance between the selected indices from each route is limited to 4. That is, we only consider cuts at $\sigma^{(1)}_{(i)}$ and $\sigma^{(2)}_{(j)}$ with $|i - j| \leq 4$.
- Swap : interchange two stations taken from two different routes.

Intra-route neighborhoods are :

- 2-opt : reverse a sub-sequence of a route (see Figure 11). The sub-sequence length is limited to 4.
- Reinsert : select a sub-sequence and reinsert it somewhere else in the route; only a random sample of the neighborhood is considered.

The cost function evaluations during the search are computed in $O(1)$ time for all neighborhoods using the auxiliary data structures and route sequence concatenation formulas given in appendix A.

4.6 Auxiliary Data Structures Update

We store matrices that contain $q_{min}(\sigma_{ij})$ and $q_{max}(\sigma_{ij})$ for all sub-routes of the initial route in order to evaluate the solution cost in constant time. Initially and during solution updates, we construct (or reconstruct) these two matrices through recurrence formulas. First, the diagonal of the matrix $q_{min}(\sigma_{ij})$ is filled with the formula below

$$q_{min}(\sigma_{ii}) = \min(0, q_i).$$

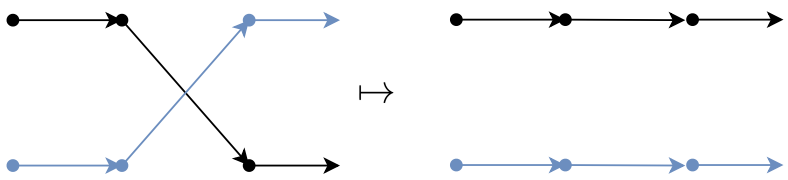


Figure 10: 2-opt*: Two routes are cut and reconnected with the other's ending segment.

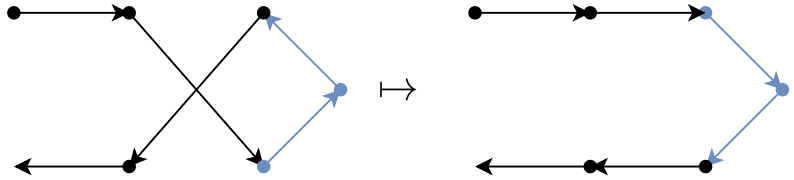


Figure 11: 2-opt: The blue segment in the route is reversed.

Then, the recurrence formula for $q_{min}(\sigma_{ij})$ is:

$$q_{min}(\sigma_{ij}) = \min(q_{min}(\sigma_{ij-1}), q_{sum}(\sigma_{i,j})) \; for \; 0 \leq i \leq j \leq n$$

Note that we obtain the $q_{max}(\sigma_{ij})$ matrix by just replacing minimum by maximum. This update procedure can be computed in $O(n^2)$ time with n the number of nodes in the route. Also, with $q_{sum}(\{\}) = 0$, the recurrence formula for $q_{sum}(\sigma_{0,i})$ is simply given by:

$$q_{sum}(\sigma_{0,i}) = q_{sum}(\sigma_{0,i-1}) + q_{\sigma_{(i)}}$$

Finally, $q_{sum}(\sigma_{i,j})$ can be retrieved with $q_{sum}(\sigma_{i,j}) = q_{sum}(\sigma_{0,j}) - q_{sum}(\sigma_{0,i-1})$.

5 Case Study

Here, we compute solutions to the static relocation problem for Montreal's BIXI bike sharing system. We use $\gamma_Q = 0.5$, $\gamma_T = 2.0$ for local search and $\gamma_Q = 0.3$ during the shaking stage. We optimize relocation quantities for the morning period from $t_0 =$ 5h to $t_1 =$ 11h. We compute plans for relocation operations during the night while ignoring the traffic impact. We also use a 4:1 ratio for time travel cost in minutes vs distance travel cost in kms ($\alpha = 0.8$ and $\beta = 0.2$, see 2.1; note that if $\alpha = 0$ and $\beta = 1$ we have the classic route distance minimization). The handling time per bike is fixed at 30 seconds (with 2 workers) and initial handling time per station is $\delta_0 = 1$ minute. Finally, truck capacity is $C = 50$ and time limit is $T = 7h$. Figures 12, 13 and 14 show some computed solutions (max. 90s computation time) for days taken from the test set. Additional explanations are provided in the caption of each figure.

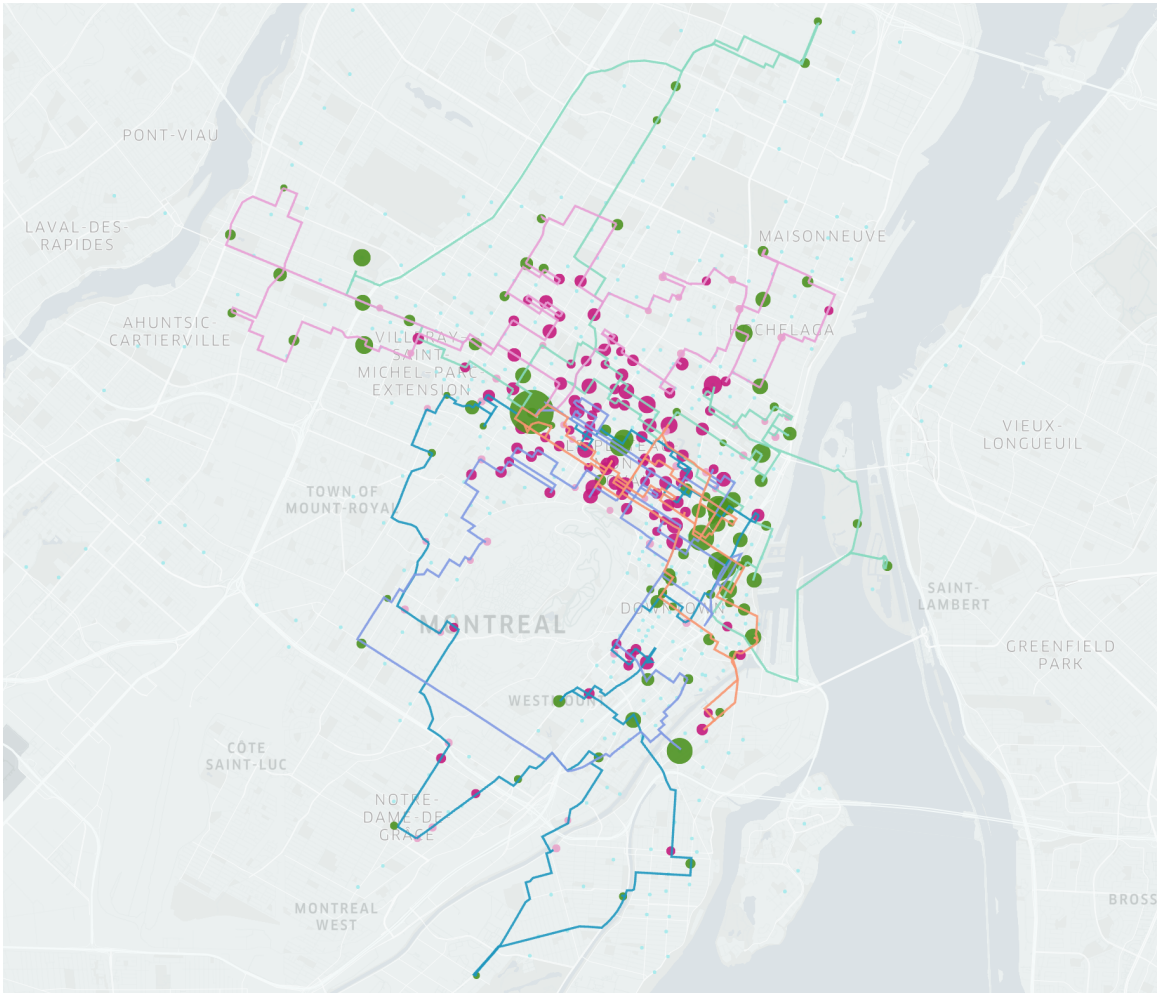


Figure 12: 5-truck solution for April 17, 2019, a sunny Wednesday (9°C at 10 a.m.). Red-pink dots represents stations where we deliver bike and green dots represent station to pick-up from. The size of the radius corresponds to the number of bikes to be picked up or delivered. We relocate 962 bikes (around 13 % of all bikes) affecting 239 stations. Being a moderately cold day of April, demand for bikes is not at its peak, with around 20 000 rides. It stands close to 1/2 of peak summer day ridership. Note that relocation empties stations in the downtown area (in green) to fill up stations in more residential-like neighborhoods such as the Plateau (in red-pink). This is to be expected. Also, note that the biggest green dot located in the center-left is the depot. We used $\theta_0 = 0.8$. Travel cost: 619. The objective cost was reduced by 8.9% and the adjusted evaluation cost was reduced by 13.6% from our initial solution.

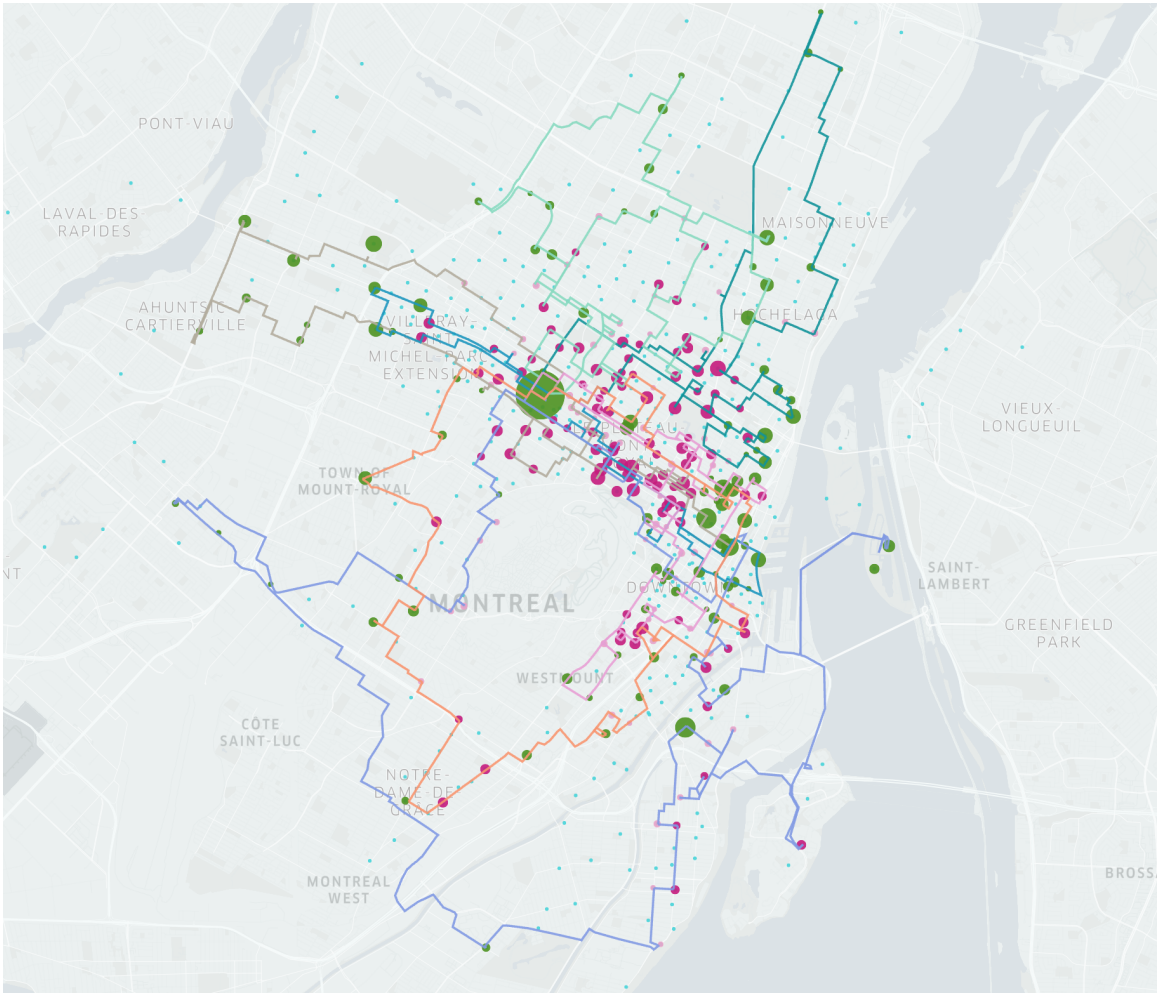


Figure 13: Full rebalancing 7-truck solution for July 12, 2019, a cloudy Friday (23°C at 10 a.m.). We relocate 1245 bikes (around 17 % of all bikes) affecting 263 stations. This hot summer day corresponds to peak demand in bikes with more than 37 000 rides. We used $\theta_0 = 0.8$. Travel cost: 721. The objective cost was reduced by 11.0% and the adjusted evaluation cost was reduced by 13.8% from our initial solution.

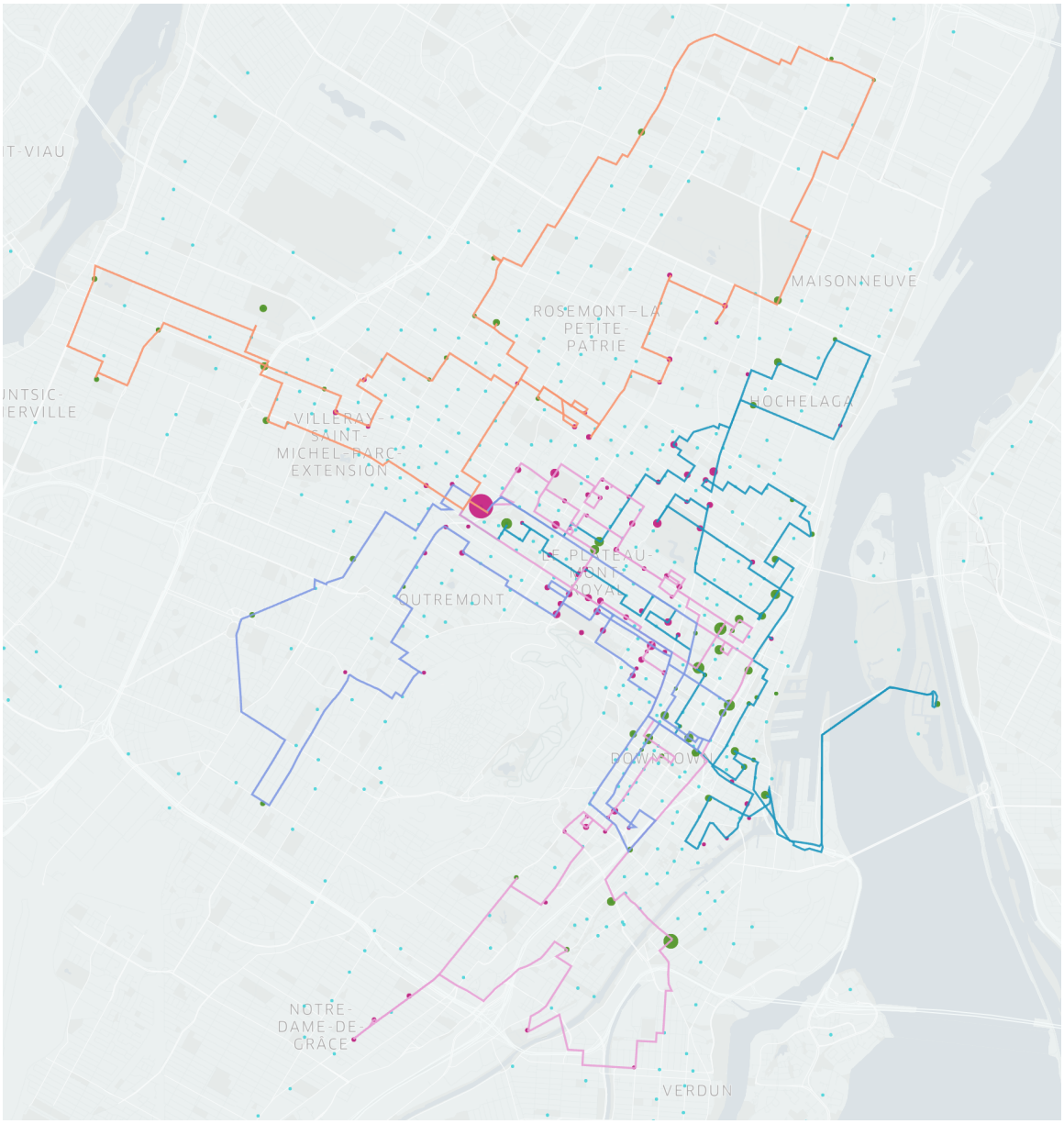


Figure 14: Light rebalancing 4-truck solution for July 12, 2019, a cloudy Friday (23°C at 10h). We relocate 560 bikes (around 8 % of all bikes) affecting 160 stations. We used $\theta_0 = 0.2$. Travel cost: 474. The objective cost was reduced by 6.0% and the adjusted evaluation cost was reduced by 15.2% from our initial solution. This result demonstrates the algorithm's flexibility with regard to the number of relocations in the final solution, which can be parameterized by θ_0 and n in Algorithm 1.

6 Conclusion

Our approach was successful at finding good relocation quantities and solving the vehicle routing problem. It was also successful at focusing only on the important subset of stations that are affected, by taking into account relocated arrivals at empty stations and relocated departures at full stations through simulation and by only minimizing those quantities. Further work needs to be done to adjust our work for the dynamic relocation problem. Especially, by considering demand prediction and route optimization as a whole, instead of computing first estimations of relocation quantities and second vehicle routes, better solutions should be obtained.

References

[1] T. Bulhões, A. Subramanian, G. Erdoğan, and G. Laporte. The static bike relocation problem with multiple vehicles and visits. *European Journal of Operational Research*, 264(2):508–523, 2018.

[2] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[3] M. Gendreau and J.-Y. Potvin. *Tabu Search*, pages 41–59. Springer US, Boston, MA, 2010.

[4] P. Hansen, N. Mladenović, J. Brimberg, and José A. M. Pérez. *Variable Neighborhood Search*, pages 61–86. Springer US, Boston, MA, 2010.

[5] Sin C. Ho and W.Y. Szeto. Solving a static repositioning problem in bike-sharing systems using iterated tabu search. *Transportation Research Part E: Logistics and Transportation Review*, 69:180–198, 2014.

[6] A. Kabra, E. Belavina, and K. Girotra. Bike-share systems: Accessibility and availability. *Management Science*, 66(9):3803–3824, 2020.

[7] M. Benchimol, P. Benchimol, B. Chappert, A. de la Taille, F. Laroche, F. Meunier, and L. Robinet. Balancing the stations of a self service "bike hire" system. *RAIRO-Oper. Res.*, 45(1):37–61, 2011.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] Y. Ren, L. Meng, F. Zhao, C. Zhang, H. Guo, Y. Tian, W. Tong, and J. W. Sutherland. An improved general variable neighborhood search for a static bike-sharing rebalancing problem considering the depot inventory. *Expert Systems with Applications*, 160:113752, 2020.

[10] S. Ross. Chapter 4 - Generating discrete random variables. In S. Ross, editor, *Simulation*, pages 47–68. Academic Press, Fifth edition, 2013.

[11] C.S. Shui and W.Y. Szeto. A review of bicycle-sharing service planning problems. *Transportation Research Part C: Emerging Technologies*, 117:102648, 2020.

A Appendix

This appendix section provides constant time formulas for computing $c(\sigma'), t_{tot}(\sigma'), q_{min}(\sigma')$ and $q_{max}(\sigma')$, which are required for adjusted cost function evaluations during the exploration of a neighborhood.

We compute $q_{min}(\sigma')$ and $q_{max}(\sigma')$ using concatenation identities given by [1]. That is :

$$q_{min}(\sigma_1 \oplus \sigma_2) = \min(q_{min}(\sigma_1), q_{sum}(\sigma_1) + q_{min}(\sigma_2))$$
$$q_{max}(\sigma_1 \oplus \sigma_2) = \max(q_{max}(\sigma_1), q_{sum}(\sigma_1) + q_{max}(\sigma_2))$$

with $\oplus$ the route concatenation operator.

A.1 2-opt Cost Variation

We define 2-opt operations on a route with i, j respectively the first and last node of the reversed segment (with $|\sigma| = n$) :

$$\text{2-opt}(\sigma, i, j) = \sigma' = \sigma_{0,i-1} \oplus \overleftarrow{\sigma_{i,j}} \oplus \sigma_{j+1,n-1}$$

We can compute the new cost in $O(1)$ time by using the following formulas :

$$c(\sigma') = c(\sigma_{0,i-1}) + c_{\sigma(i-1),\sigma(j)} + c(\overleftarrow{\sigma_{i,j}}) + c_{\sigma(i),\sigma(j+1)} + c(\sigma_{j+1,n-1})$$

$$t_{tot}(\sigma') = t_{tot}(\sigma_{0,i-1}) + t_{\sigma(i-1),\sigma(j)} + t_{tot}(\overleftarrow{\sigma_{i,j}}) + t_{\sigma(i),\sigma(j+1)} + t_{tot}(\sigma_{j+1,n-1})$$

We start by computing q_{min}, q_{max} and q_{sum} for segment $\overleftarrow{\sigma_{i,j}}$

$$q_{min}(\overleftarrow{\sigma_{ij}}) = q_{sum}(\sigma_{i,j}) - q_{max}(\sigma_{i,j})$$

$$q_{max}(\overleftarrow{\sigma_{ij}}) = q_{sum}(\sigma_{i,j}) - q_{min}(\sigma_{i,j}$$

$$q_{sum}(\overleftarrow{\sigma_{i,j}}) = q_{sum}(\sigma_{i,j})$$

Then, we compute q_{min}, q_{max} and q_{sum} for $\sigma_{0,i-1} \oplus \overleftarrow{\sigma_{i,j}}$:

$$q_{min}(\sigma_{0,i-1} \oplus \overleftarrow{\sigma_{ij}}) = \min(q_{min}(\sigma_{0,i-1}), q_{sum}(\sigma_{0,i-1}) + q_{min}(\overleftarrow{\sigma_{i,j}}))$$

$$q_{max}(\sigma_{0,i-1} \oplus \overleftarrow{\sigma_{ij}}) = \max(q_{max}(\sigma_{0,i-1}), q_{sum}(\sigma_{0,i-1}) + q_{max}(\overleftarrow{\sigma_{i,j}}))$$

$$q_{sum}(\sigma_{0,i-1} \oplus \overleftarrow{\sigma_{i,j}}) = q_{sum}(\sigma_{0,i-1}) + q_{sum}(\overleftarrow{\sigma_{i,j}}) = q_{sum}(\sigma_{0,j})$$

And finally, we compute q_{min} and q_{max} for route σ':

$$q_{min}(\sigma_{0,i-1} \oplus \overleftarrow{\sigma_{i,j}} \oplus \sigma_{j+1,n-1}) = \min(q_{min}(\sigma_{0,i-1} \oplus \overleftarrow{\sigma_{ij}}), q_{sum}(\sigma_{0,j}) + q_{min}(\sigma_{j+1,n-1}))$$

$$q_{max}(\sigma_{0,i-1} \oplus \overleftarrow{\sigma_{i,j}} \oplus \sigma_{j+1,n-1}) = \max(q_{max}(\sigma_{0,i-1} \oplus \overleftarrow{\sigma_{ij}}), q_{sum}(\sigma_{0,j}) + q_{max}(\sigma_{j+1,n-1}))$$

With these formulas, we can compute $q_{min}(\sigma')$ and $q_{max}(\sigma')$ in $O(1)$ time.

A.2 Shift Cost Variation

Let $\sigma^{(1)}$ and $\sigma^{(2)}$ two routes of length n_1 and n_2, respectively. When we apply Shift$(\sigma^{(1)}, \sigma^{(2)}, i, j)$, the result is :

$$\sigma^{'(1)} = \sigma^{(1)}_{0,i-1} \oplus \sigma^{(1)}_{i+1,n_1-1}$$

$$\sigma^{'(2)} = \sigma^{(2)}_{0,j} \oplus \sigma^{(1)}_{i} \oplus \sigma^{(2)}_{j+1,n_2-1}$$

The updates are :

$$c(\sigma^{'(1)}) = c(\sigma^{(1)}_{0,i-1}) + c_{\sigma^{(1)}(i-1),\sigma^{(1)}(i+1)} + c(\sigma^{(1)}_{i+1,n_1-1})$$

$$t_{tot}(\sigma^{'(1)}) = t_{tot}(\sigma^{(1)}_{0,i-1}) + t_{\sigma^{(1)}(i-1),\sigma^{(1)}(i+1)} + t_{tot}(\sigma^{(1)}_{i+1,n_1-1})$$

$$c(\sigma^{'(2)}) = c(\sigma^{(2)}_{0,j}) + c_{\sigma^{(2)}(j),\sigma^{(1)}(i)} + c_{\sigma^{(1)}(i),\sigma^{(2)}(j+1)} + c(\sigma^{(2)}_{j+1,n_2-1})$$

$$t_{tot}(\sigma^{'(2)}) = t_{tot}(\sigma^{(2)}_{0,j}) + t_{\sigma^{(2)}(j),\sigma^{(1)}(i)} + \delta_0 + \delta|q_i| + t_{\sigma^{(1)}(i),\sigma^{(2)}(j+1)} + t_{tot}(\sigma^{(2)}_{j+1,n_2-1})$$

$$q_{sum}(\sigma^{'(1)}) = q_{sum}(\sigma^{(1)}) - q_{\sigma^{'(1)}(i)}$$

$$q_{sum}(\sigma^{'(2)}) = q_{sum}(\sigma^{(2)}) + q_{\sigma^{'(1)}(i)}$$

$$q_{min}(\sigma^{'(1)}) = \min(q_{min}(\sigma^{(1)}_{0,i-1}), q_{sum}(\sigma^{(1)}_{0,i-1}) + q_{min}(\sigma^{(1)}_{i+1,n_1-1}))$$

$$q_{max}(\sigma^{'(1)}) = \max(q_{max}(\sigma^{(1)}_{0,i-1}), q_{sum}(\sigma^{(1)}_{0,i-1}) + q_{max}(\sigma^{(1)}_{i+1,n_1-1}))$$

$$q_{min}(\sigma^{'(2)}) = \min(q_{min}(\sigma^{(2)}_{0,j}), q_{sum}(\sigma^{(2)}_{0,j}) + \min(q^{-}_{\sigma^{'(1)}(i)}, q_{\sigma^{'(1)}(i)} + q_{min}(\sigma^{(2)}_{i+1,n_2-1})))$$

$$q_{max}(\sigma^{'(2)}) = \max(q_{max}(\sigma^{(2)}_{0,j}), q_{sum}(\sigma^{(2)}_{0,j}) + \max(q^{+}_{\sigma^{'(1)}(i)}, q_{\sigma^{'(1)}(i)} + q_{max}(\sigma^{(2)}_{i+1,n_2-1})))$$

A.3 Swap Cost Variation

Let us consider the function Swap$(\sigma^{(1)}, \sigma^{(2)}, i, j)$, where $\sigma^{(1)}$ and $\sigma^{(2)}$ are two routes of length n_1 and n_2, respectively, and i, j are indexes within $\sigma^{(1)}$ and $\sigma^{(2)}$, respectively, of the chosen stations (nodes) to be swapped.

The new updated routes are:

$$\sigma^{'(1)} = \sigma^{(1)}_{0,i-1} \oplus \sigma^{(2)}_{j} \oplus \sigma^{(1)}_{i+1,n_1-1}$$

$$\sigma'^{(2)} = \sigma^{(2)}_{0,j-1} \oplus \sigma^{(1)}_{i} \oplus \sigma^{(2)}_{j+1,n_2-1}$$

The updates are:

$$c(\sigma'^{(1)}) = c(\sigma^{(1)}_{0,i-1}) + c_{\sigma^{(1)}(i-1),\sigma^{(2)}(j)} + c_{\sigma^{(2)}(j),\sigma^{(1)}(i+1)} + c(\sigma^{(1)}_{i+1,n_1-1})$$
$$c(\sigma'^{(2)}) = c(\sigma^{(2)}_{0,j-1}) + c_{\sigma^{(2)}(j-1),\sigma^{(1)}(i)} + c_{\sigma^{(1)}(i),\sigma^{(2)}(j+1)} + c(\sigma^{(2)}_{j+1,n_2-1})$$

$$q_{sum}(\sigma'^{(1)}) = q_{sum}(\sigma^{(1)}) - q_{\sigma'^{(1)}(i)} + q_{\sigma'^{(2)}(j)}$$
$$q_{sum}(\sigma'^{(2)}) = q_{sum}(\sigma^{(2)}) - q_{\sigma'^{(2)}(j)} + q_{\sigma'^{(1)}(i)}$$

$$q_{min}(\sigma'^{(1)}) = q_{min}(\sigma^{(1)}_{0,i-1} \oplus (\sigma^{(2)}_{j} \oplus \sigma^{(1)}_{i+1,n_1-1}))$$
$$= \min(q_{min}(\sigma^{(1)}_{0,i-1}), q_{sum}(\sigma^{(1)}_{0,i-1}) + \min(q^{-}_{\sigma'^{(2)}(j)}, q_{\sigma'^{(2)}(j)} + q_{min}(\sigma^{(1)}_{i+1,n_1-1})))$$
$$q_{min}(\sigma'^{(2)}) = q_{min}(\sigma^{(2)}_{0,j-1} \oplus (\sigma^{(1)}_{i} \oplus \sigma^{(2)}_{j+1,n_2-1}))$$
$$= \min(q_{min}(\sigma^{(2)}_{0,j-1}), q_{sum}(\sigma^{(2)}_{0,j-1}) + \min(q^{-}_{\sigma'^{(1)}(i)}, q_{\sigma'^{(1)}(i)} + q_{min}(\sigma^{(2)}_{j+1,n_2-1})))$$

$$q_{max}(\sigma'^{(1)}) = q_{max}(\sigma^{(1)}_{0,i-1} \oplus (\sigma^{(2)}_{j} \oplus \sigma^{(1)}_{i+1,n_1-1}))$$
$$= \max(q_{max}(\sigma^{(1)}_{0,i-1}), q_{sum}(\sigma^{(1)}_{0,i-1}) + \max(q^{+}_{\sigma'^{(2)}(j)}, q_{\sigma'^{(2)}(j)} + q_{max}(\sigma^{(1)}_{i+1,n_1-1})))$$
$$q_{max}(\sigma'^{(2)}) = q_{max}(\sigma^{(2)}_{0,j-1} \oplus (\sigma^{(1)}_{i} \oplus \sigma^{(2)}_{j+1,n_2-1}))$$
$$= \max(q_{max}(\sigma^{(2)}_{0,j-1}), q_{sum}(\sigma^{(2)}_{0,j-1}) + \min(q^{+}_{\sigma'^{(1)}(i)}, q_{\sigma'^{(1)}(i)} + q_{max}(\sigma^{(2)}_{j+1,n_2-1})))$$

$$t_{tot}(\sigma'^{(1)}) = t_{tot}(\sigma^{(1)}_{0,i-1}) + t_{\sigma^{(1)}(i-1),\sigma^{(2)}(j)} + \delta_0 + \delta|q_j| + t_{\sigma^{(2)}(j),\sigma^{(1)}(i+1)} + t_{tot}(\sigma^{(1)}_{1+1,n_1-1})$$

$$t_{tot}(\sigma'^{(2)}) = t_{tot}(\sigma^{(2)}_{0,j-1}) + t_{\sigma^{(2)}(j-1),\sigma^{(1)}(i)} + \delta_0 + \delta|q_i| + t_{\sigma^{(1)}(i),\sigma^{(2)}(j+1)} + t_{tot}(\sigma^{(2)}_{j+1,n_2-1})$$

Finally, with all the information above, we can easily assess our new values of the adjusted cost functions $Z(\sigma'^{(1)})$ and $Z(\sigma'^{(2)})$.

A.4 2-opt* Cost Variation

We have 2-opt*, an inter-route neighborhood that takes two routes $\sigma^{(1)}$ and $\sigma^{(2)}$ and two stations $0 < i < n$, $0, < j < m$, one from each route. The first route and the second route are cut in two at stations i and j respectively. The two routes can be rewritten in the following way:

$$\sigma^{(1)} = \sigma^{(1)}_{0,i-1} \oplus \sigma^{(1)}_{i,n}$$

$$\sigma^{(2)} = \sigma^{(2)}_{0,j-1} \oplus \sigma^{(2)}_{j,m}$$

Then, the first part of the first route is concatenated with the second part of the second route and the first part of the second route is concatenated with the second part of the first route. This gives the following new routes:

$$\sigma^{(1)'} = \sigma^{(1)}_{0,i-1} \oplus \sigma^{(2)}_{j,m}$$

$$\sigma^{(2)'} = \sigma^{(2)}_{0,j-1} \oplus \sigma^{(1)}_{i,n}$$

We can calculate $q_{min}(\sigma^{(1)'})$ and $q_{max}(\sigma^{(1)'})$ with the following formulas:

$$q_{min}(\sigma^{(1)'}) = \min(q_{min}(\sigma^{(1)}_{0,i-1}), q_{sum}(\sigma^{(1)}_{0,i-1}) + q_{min}(\sigma^{(2)}_{j,m}))$$

$$q_{max}(\sigma^{(1)'}) = \max(q_{max}(\sigma^{(1)}_{0,i-1}), q_{sum}(\sigma^{(1)}_{0,i-1}) + q_{max}(\sigma^{(2)}_{j,m}))$$

We can calculate the cost objective and total time with formulas:

$$c(\sigma^{(1)'}) = c(\sigma^{(1)}_{0,i-1}) + c_{\sigma^{(1)}(i-1),\sigma^{(1)}(j)} + c(\sigma^{(2)}_{j,m})$$

$$t_{tot}(\sigma^{(1)'}) = t_{tot}(\sigma^{(1)}_{0,i-1}) + t_{\sigma^{(1)}(i-1),\sigma^{(1)}(j)} + t_{tot}(\sigma^{(2)}_{j,m})$$

We note that the formulas are similar for route $\sigma^{(2)'}$.

A.5 Reinsert Cost Variation

Let us consider the function Reinsert(σ, i, j, k), where σ is a route of length n.

• If $k \geq j + 1$:
The new updated route is:

$$\sigma^{'} = \sigma_{0,i-1} \oplus \sigma_{j+1,k} \oplus \sigma_{i,j} \oplus \sigma_{k+1,n-1}$$

The updates are:

$$c(\sigma^{'}) = c(\sigma_{0,i-1}) + c_{\sigma(i-1),\sigma(j+1)} + c(\sigma_{j+1,k}) + c_{\sigma(k),\sigma(i)} + c(\sigma_{i,j}) + c_{\sigma(j),\sigma(k+1)} + c(\sigma_{k+1,n-1})$$

$$q_{sum}(\sigma^{'}) = q_{sum}(\sigma)$$

$$q_{min}(\sigma^{'}) = q_{min}(\sigma_{0,i-1} \oplus \sigma_{j+1,k} \oplus \sigma_{i,j} \oplus \sigma_{k+1,n-1})$$
$$= \min(q_{min}(\sigma_{0,i-1}), q_{sum}(\sigma_{0,i-1}) + \min(q_{min}(\sigma_{j+1,k}), q_{sum}(\sigma_{j+1,k}) +$$
$$\min(q_{min}(\sigma_{i,j}), q_{sum}(\sigma_{i,j}) + q_{min}(\sigma_{k+1,n-1})))$$

$$q_{max}(\sigma^{'}) = q_{max}(\sigma_{0,i-1} \oplus \sigma_{j+1,k} \oplus \sigma_{i,j} \oplus \sigma_{k+1,n-1})$$
$$= \max(q_{max}(\sigma_{0,i-1}), q_{sum}(\sigma_{0,i-1}) + \max(q_{max}(\sigma_{j+1,k}), q_{sum}(\sigma_{j+1,k}) +$$

$$\max(q_{max}(\sigma_{i,j}), q_{sum}(\sigma_{i,j}) + q_{max}(\sigma_{k+1,n-1})))$$

$$t_{tot}(\sigma^{'}) = t_{tot}(\sigma_{0,i-1}) + t_{\sigma(i-1),\sigma(j+1)} + t_{tot}(\sigma_{j+1,k}) + t_{\sigma(k),\sigma(i)} + t_{tot}(\sigma_{i,j}) + t_{\sigma(j),\sigma(k+1)} + t_{tot}(\sigma_{k+1,n-1})$$

• If $0 \leq k \leq i-2$:
The new updated route is:

$$\sigma^{'} = \sigma_{0,k-1} \oplus \sigma_{i,j} \oplus \sigma_{k+1,i-1} \oplus \sigma_{j+1,n-1}$$

The updates are:

$$c(\sigma^{'}) = c(\sigma_{0,k-1}) + c_{\sigma(k-1),\sigma(i)} + c(\sigma_{i,j}) + c_{\sigma(j),\sigma(k)} + c(\sigma_{k,i-1}) + c_{\sigma(i-1),\sigma(j+1)} + c(\sigma_{j+1,n-1})$$

$$q_{sum}(\sigma^{'}) = q_{sum}(\sigma)$$

$$q_{min}(\sigma^{'}) = q_{min}(\sigma_{0,k-1} \oplus \sigma_{i,j} \oplus \sigma_{k,i-1} \oplus \sigma_{j+1,n-1})$$
$$= \min(q_{min}(\sigma_{0,k-1}), q_{sum}(\sigma_{0,k-1}) + \min(q_{min}(\sigma_{i,j}), q_{sum}(\sigma_{i,j}) +$$
$$\min(q_{min}(\sigma_{k,i-1}), q_{sum}(\sigma_{k,i-1}) + q_{min}(\sigma_{j+1,n-1})))$$

$$q_{max}(\sigma^{'}) = q_{max}(\sigma_{0,k-1} \oplus \sigma_{i,j} \oplus \sigma_{k,i-1} \oplus \sigma_{j+1,n-1})$$
$$= \max(q_{max}(\sigma_{0,k-1}), q_{sum}(\sigma_{0,k-1}) + \max(q_{max}(\sigma_{i,j}), q_{sum}(\sigma_{i,j}) +$$
$$\max(q_{max}(\sigma_{k,i-1}), q_{sum}(\sigma_{k,i-1}) + q_{max}(\sigma_{j+1,n-1})))$$

$$t_{tot}(\sigma^{'}) = t_{tot}(\sigma_{0,k-1}) + t_{\sigma(k-1),\sigma(i)} + t_{tot}(\sigma_{i,j}) + t_{\sigma(j),\sigma(k)} + t_{tot}(\sigma_{k,i-1}) + t_{\sigma(i-1),\sigma(j+1)} + t_{tot}(\sigma_{j+1,n-1})$$

A.6 Add Cost Variation

The update rules defined here are used in the construction of the initial solution. We add node s after σ_i.

$$\sigma' = \sigma_{0,i} \oplus s \oplus \sigma_{i+1,n_1-1}$$

The updates are :

$$c(\sigma') = c(\sigma_{0,i}) + c_{\sigma_{(i)},s} + c_{s,\sigma_{(i+1)}} + c(\sigma_{i+1,n_1-1})$$

$$t_{tot}(\sigma') = t_{tot}(\sigma_{0,i}) + t_{\sigma_{(i)},s} + \delta_0 + \delta|q_s| + t_{s,\sigma_{(i+1)}} + t_{tot}(\sigma_{i+1,n_1-1})$$

$$q_{sum}(\sigma') = q_{sum}(\sigma) + q_s$$

$$q_{min}(\sigma') = \min(q_{min}(\sigma_{0,i}), q_{sum}(\sigma_{0,i}) + \min(\min(0, q_s), q_s + q_{min}(\sigma_{i+1,n_1-1})))$$

$$q_{max}(\sigma') = \max(q_{max}(\sigma_{0,i}), q_{sum}(\sigma_{0,i}) + \max(\max(0, q_s), q_s + q_{max}(\sigma_{i+1,n_1-1})))$$