Bicycle Improvement Design Problem using real bike trajectories

Félix Repusseau[1], Tifenn Rault[1], and Emmanuel Néron[1]

[1]La Compagnie des Mobilités
[2]Laboratoire d'informatique fondamentale et Appliquée de Tours, Polytech Tours 64 av. Jean Portalis, F-37200 Tours

Abstract

While the development of mobility with low GHG emissions is an essential issue for all cities, it now appears necessary to intelligently deploy bicycle facilities to strengthen users' feelings of safety. The problem we are addressing here consists in using a set of GPS traces corresponding to paths taken by cyclists, to propose improvements to the cycling network, eg. installation of cycle lanes on one or more sections, in order to increase the overall safety of users, respecting a given budget. Our contribution is to propose an Integer Linear Programming formulation that can be used on small size instances, and some heuristics that can be used to get proposition of network improvements on real size instances.

1 Introduction and literature review

After having proposed methods to compute secure paths dedicated to bicycle in order to improve safety for bike users [2], we are interested here in the improvement of the cycling network in order to increase the overall safety of the users.

2 Model and Integer Linear Programming Formulation

2.1 Model Overview

Let us recall briefly the problem that is addressed :

- A graph describing the network, i.e, cycling and non cycling arcs
- For the set of arcs that belong to non cycling network, we get costs representing the distance and the unsafety of the roads, with and without improvements (cf. fig. 1) ;
- For the arcs that belong to cycling network, we get costs representing the distance and the unsafety of the roads
- An overall budget. Here we consider that the cost of an improvement is proportional to the distance of the arc that is modified. Thus the budget can be expressed in meters.
- A set of cyclist's GPS traces, including an origin/destination (OD) pair and user preference between distance and unsafety 2. The user preference is obtained by comparing the cyclist's trajectory to the nearest Pareto solution obtained using a label setting algorithm ([2]) between O and D.
- Our aim is to determine the set of arcs that have to be improved to get the maximum user satisfaction, respecting the overall budget.

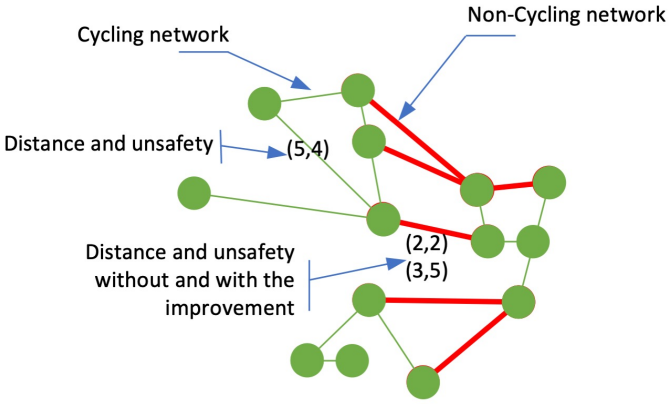

Figure 1: Initial network

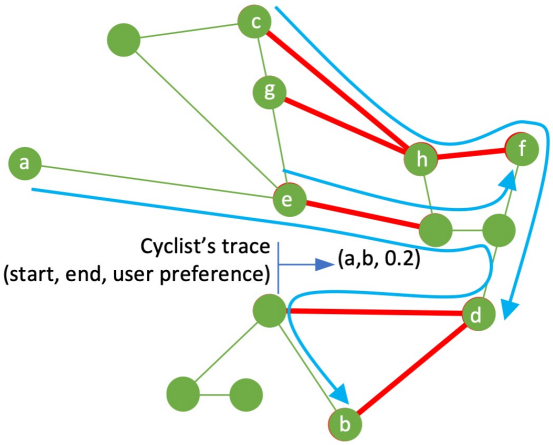

Figure 2: Initial network and cyclists' traces

The two figures 1 and 2 present an instance. Our aim is to determine the set of arc of the non-cycling sub-graph that have to be improved, i.e., to become a part of the cycling sub-graph. Notice that on this instance depending on, costs on the arcs, the budget and the user preferences, it may be interesting to improve the arc (g, h), even if it does not belong to a trajectory, such that the path (e, f) and the (c, d) are modified into (e, g, h, f) and (c, g, h, f, d).

2.2 Complexity

Proposition 1 *The problem is $\mathcal{NP}$-Hard in the strong sense.*

Demonstration . Demo redigée par Tifenn

2.3 ILP formulation

The main idea of the ILP formulation id to decide what are the arc on the non cycling network that have to be improved, respecting the overall budget. Once the arcs to be modified are determined the path corresponding to cyclists' traces on the modified network are recomputed. Our aim is that the global user satisfaction is maximized (depending on its own preference between distance and unsafety).

Parameters:

- The network is represented by an oriented graph $G(X, A)$ with X ($|X| = n$) the set of nodes, and A the set of arcs. for seek of simplicity let $succ(i)$ denotes the successors of node $i \in X$, and $pred(i)$ denotes the predecessors of node $i \in G$;
- A is partitioned in two sets : the cycling arcs A_c and the non cycling arcs A_n
- Each arc $(i, j) \in A_n$ is associated to four costs : c^1_{ij} (resp. $\overline{c_{ij}}^1$) is the distance cost of arc (i, j) before (resp. after) improvements, and c^2_{ij} (resp. $\overline{c_{ij}}^2$) is the unsafety cost of arc (i, j) before (resp. after) improvements.
- Each arc $(i, j) \in A_c$ is associated to two costs : c^1_{ij} is the distance cost of arc (i, j) and c^2_{ij} is the unsafety cost of arc (i, j).
- Let P ($|P| = s$) denotes the cyclists' GPS traces in what follows. Each trace $k \in 1, .., s$ has been matched onto the graph and converted into a path. Then it corresponds to a path in G between an origin node $start_k \in X$, a destination node $end_k \in X$, and a weight $\alpha_k \in [0..1]$ representing the user's preference in terms of distance versus unsafety.
- B is the overall budget that cannot be exceeded.

Variables:

- The main decision variables are the variables $\overline{\sigma_{ij}}$, which is equal to 1 if arc $(i, j) \in A_n$ is modified, 0 otherwise. When this variable is equal to one, (i, j) must be modified for all paths going through it. Notice that for arcs in A_c whose already belong to the cycling network, which means they are already improved, the corresponding $\overline{\sigma_{ij}}$ can be considered as fixed to 0
- Some variables are used to model the routing of bikers : δ^k_{ij} is equal to 1 if arc (i, j) is in track k (modified or not).
- Finally, to compute the cost of each path, we need the two following variables :
  - y^k_{ij} : $y^k_{ij} = 1$ if arc $(i, j) \in A$ is part of the path k and the arc (i, j) is not modified, 0 otherwise.
  - $\overline{y_{ij}}^k$: $\overline{y_{ij}}^k = 1$ if arc $(i, j) \in A$ is part of the path k and the arc (i, j) is modified, 0 otherwise.

Cost Function:

- Once the variables are set, we know the effective distance and unsafety for each arc. Then we are able to compute for each cyclist's trace a shortest path that minimizes the linear combination of the distance and unsafety regarding the user preference attached to each to trajectory. This cost is denoted by $C_k \forall k \in P$
- The cost function aims to minimize the overall costs for all the users.

$$Minimize \sum_{k \in P} C_k \tag{1}$$

$$y_{ij}^k + \overline{y_{ij}}^k \leq 1, \forall k \in P, \forall (i,j) \in A \tag{2}$$

$$\overline{\sigma_{ij}} \geq \overline{y_{ij}}^k, \forall k \in P, \forall (i,j) \in A \text{ and } 1 - \overline{\sigma_{ij}} \geq y_{ij}^k, \forall k \in P, \forall (i,j) \in A \tag{3}$$

$$\delta_{ij}^k = y_{ij}^k + \overline{y_{ij}}^k, \forall (i,j) \in A, \forall k \in P, \tag{4}$$

$$\sum_{j \in succ(start_k)} \delta_{start_k,j}^k = 1, \forall k \in P, \text{ and } \sum_{j \in pred(end_k)} \delta_{j,end_k}^k = 1, \forall k \in P, \tag{5}$$

$$\sum_{i \in pred(j)} \delta_{i,j}^k = \sum_{l \in succ(j)} \delta_{j,l}^k, \forall k \in P, \forall j \in N \backslash \{start_k, end_k\} \tag{6}$$

$$\sum_{i \in pred(j)} \delta_{i,j}^k \leq 1, \forall k \in P, \forall j \in N \text{ and } \sum_{l \in succ(j)} \delta_{j,l}^k \leq 1, \forall k \in P, \forall j \in N \tag{7}$$

$$\sum_{(i,j) \in A} \delta_{i,j}^k (c_{ij}^1 y_{ij}^k + \overline{c_{ij}}^1 \overline{y_{ij}}^k) \leq Cinit_k^1, \forall k \in P \text{ and } \sum_{(i,j) \in A} \delta_{i,j}^k (c_{ij}^2 y_{ij}^k + \overline{c_{ij}}^2 \overline{y_{ij}}^k) \leq Cinit_k^2, \forall k \in P \tag{8}$$

$$\sum_{(i,j) \in A} \overline{\sigma_{ij}} c_{ij}^1 \leq B \tag{9}$$

$$C_k = \sum_{(i,j) \in A} y_{ij}^k [(c_{ij}^1 \alpha_k) + (c_{ij}^2 (1 - \alpha_k))] + \sum_{(i,j) \in A} \overline{y_{ij}}^k [(\overline{c_{ij}}^1 \alpha_k) + (\overline{c_{ij}}^2 (1 - \alpha_k))] \tag{10}$$

(2) and (3) ensure consistency of y and $\overline{y}$ variables. These constraints ensure that if an arc is modified, it is modified for all paths passing by this arc. Alternatively, if an arc is not modified, it is not modified in other paths. Notice that (2) is not mandatory due to (3) formulation. (4) to (7) guarantee the consistency of the calculated paths between each OD pairs. (9) limit the total length of modified arcs with an upper bound B. (10) represent the cost of a path : for each path k, its cost takes into consideration whether the arcs are modified or not, as well as the user preference between distance and unsafety via the parameter α_k. Finally, the objective function 1 minimizes the sum of the costs of all the paths.

2.4 Improving the ILP resolution

some variants of the ILP formulation have been tested on small and medium size instances in order to speed up the resolution :

- **ILP0 - No improvements** : the model is given to the solver as defined previously, all the traces are given at the beginning of the resolution, no initial solution is given and the overall budget is considered.
- **ILP1 - Initial solutions is computed** : the model is given to the solver as defined previously, all the traces are given at the beginning of the resolution, but an initial solution is computed, in order to help the resolution. This initial partial solution corresponds to compute some path corresponding to the OD of each trace of T. :
  - The **Shortest paths** corresponding to all OD pairs of the traces. The underlying idea is that if the improvement of the networks correspond to the shortest path, then the improvement can be interesting for this user.
  - The **the best paths** corresponding to all OD pairs of the traces, and the user preference associated to each trace of T.

- **ILP2 - traces are added by batch**. The idea is to used the solution given by solver at an iteration with k_1 traces as the initial partial solution for solving the problem with $k_2 > k_1$ traces. In that case the overall budget is given the overall budget is considered
- **ILP3 - Budget is incrementally sets**. The idea is to used the solution given by the solver for a given budget b_1 as the initial solution for solving the problem with a budget b_2 such that $B \geq b_2 > b_1$.

3 Heuristic methods for real size instances

In this section for seek of simplicity the cycling network denotes the sub-graph in which all arcs

3.1 Arc-Frequency based heuristics

The first heuristics that are presented in this section are inspired from [**?**]. The information that is mainly used from cyclists' traces is the number of time that each arc belongs to a trace (frequency). Then arcs will be added to the cycling network in order of their decreasing frequency, if and only if the arc is adjacent to an arc that belongs to the existing cycling network. This step is repeated until there in no more candidate or the budget is completely used.

Algorithm 1 Knapsack Heuristics

Require:
Existing cycle network;
Set T set of traces a trace is a set of arcs ;
B the overall budget
Ensure:
Modified cycle network

$LC \leftarrow$ Set of arcs in the network that do not belong to the cycling network but that are adjacent to the cycling network ;
LC is sorted by decreasing order of the arc frequency in T
$Bugdet \leftarrow$ B
while $LC \neq \{\}$ and Budget > 0 **do**
  $Cand \leftarrow$ first element of LC
  Add to LC all arcs adjacent to Cand that are not in the cycling network
  Decrease Budget of Budget($Cand$)
  remove from LC all the arcs such that their budget is strictly lower than $Budget$
end while

This version is based on the strict frequency of the arc, e.g., the number of time the arcs belongs to a trace. Three different versions for ordering the arcs have been proposed :

- **Frequency** of the arcs regarding T the set of traces.
- **Average Frequency** : for each arc the frequency of the arc in T is considered, but the frequency of its predecessor and its successors with a rate equal to 0.5 is also taken into account, and the frequency of the successors of its successors (predecessors of its predecessors) with a rate equals to 0.25
- **Shortest Path Frequency**. For each trace in T we compute the shortest path form the source of the trace to the destination. The shortest path frequency for a given arc is the number of time the arc appears in a trace of T minus the number of time it appears in shortest paths set computed from T. In that case the list of candidate is sorted in the increasing order of this criteria. [`expliquer sur la base d'un exemple`]

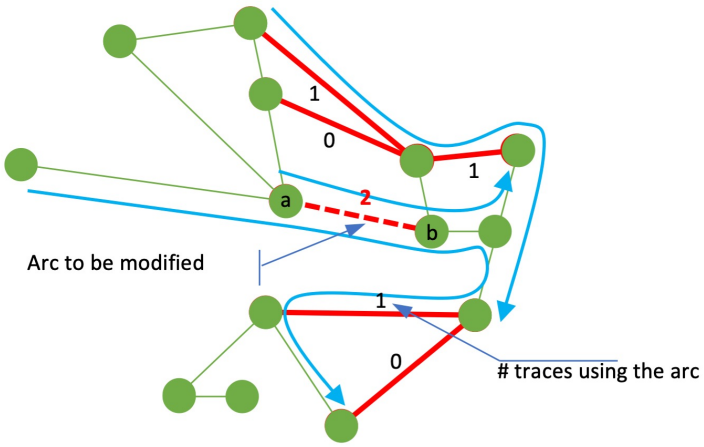

Figure 3: Arc selected by the algorithm for example of fig. 2

3.2 Graph-structure based heuristics

The next two heuristics are inspired from the ones proposed in [**?**]. The aim of this two methods is to get a connected cycling network as much as possible. In order to get connected cycling network authors proposed to connect connected components of the cycling network. These heuristics basically do not used the set of traces T, but only the cycling graph structure. An improvement to partially take into account the traces is proposed.

Algorithm 2 Connecting component

Require:
Existing cycle network;
set T set of traces ;
B the overall budget
Ensure:
Modified cycle network

Bugdet $\leftarrow B$
while $Budget > 0$ **do**
Identify all the connected component of the cycling network
$C \leftarrow$ the largest component (the one maximizing the sum of the distance if its arcs
$D \leftarrow$ the closest component from C (the one such that the minimum distance between one node in C and one node in D is minimum)
Improve the shortest path from C to D
Decrease Budget
end while

Three different versions for connecting component have been proposed :

- **Largest to closest**. For the largest connected component (the one such that the distance of its arcs is maximum) the distance to other connected component is computed (the minimum distance from all nodes of the first one to all nodes of the second one). This method aims to connect connected component that are not so far, and this make the network growing "incrementally" from its initial largest connected component.
- **Largest to second** : For the largest connected component the distance to second largest connected component is computed (the minimum distance from all nodes of the first one to all nodes of the second one). This method aims to connect connected component that are significant.
- **Largest to most used in** T. For each trace in T we decompose the set of its arcs into the sets of connected component. Then each time the initial connected component and a connected

component appears into a trace, the score of this latest is incremented. Then we connect the largest to the one is more connected to it regarding the traces in T. This one extends the notion of graph structure based heuristics in order to take into account T

3.3 Trace based heuristics

The last two methods uses more intensively set of traces that is given. The aim is to measure the impact of each arc improvement on the trace by recomputing the path respecting the users preferences between the distance and the safety. These methods used the notion of "visibility", that means the geographical area in which the cyclist may eventually change its path. The way that the visibility is computed allow to speed up the method by limiting the number of arc improvements to consider.

Algorithm 3 Arc-impact (Greedy)

Require:
Existing cycle network
set T set of traces, a trace t s given by an Origin O_t a Destination D_t and a user preference p_t;
B the overall budget
Ensure:
Modified cycle network

Bugdet $\leftarrow B$
while $Budget > 0$ **do**
for trace $t(O_t, D_t, p_t) \in T$ **do**
let A the set of arcs that belong to the visible area of t
for each arc $a \in A$ **do**
Compute the shorted path using p_t as user preference if a is improved
% (linear combination of the two criteria),
Compute the improvement cost of the new path and add it to the arc the improvement if it is non 0
end for
end for
Select the arc s with the best overall improvement,
Improve s ; decrease the budget
end while

4 Experiments

4.1 Instance Description

4.2 Results on synthetic instance

4.3 Results on real size instances

4.4 Network structure improvements

4.5 Discussion

5 Conclusion and Perspectives

References

[1] Dilkina, B., Lai, K.J., Gomes, C.P. (2011). Upgrading Shortest Paths in Networks. In: Achterberg, T., Beck, J.C. (eds) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. CPAIOR 2011. Lecture Notes in Computer Science, vol 6697. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21311-3-9

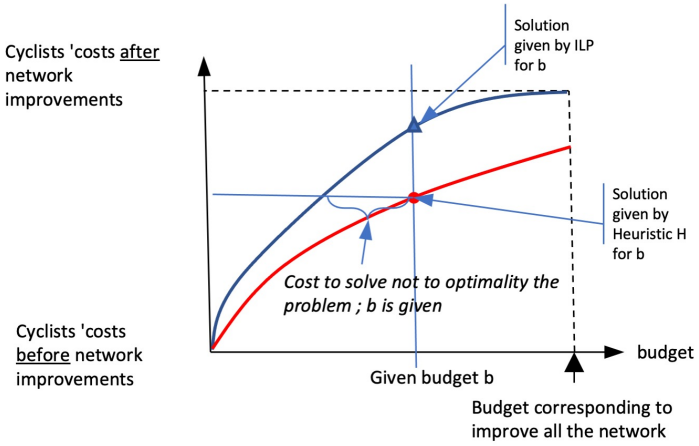

Figure 4: Kind of results 2

[2] Yannick Kergosien, Antoine Giret, Emmanuel Neron, Gaël Sauvanet. An efficient label-correcting algorithm for the multi-objective shortest path problem. *INFORMS Journal on Computing*, 2021. ⟨hal-03162962⟩

[3] Haoxiang Liu, W.Y. Szeto, Jiancheng Long, Bike network design problem with a path-size logit-based equilibrium constraint: Formulation, global optimization, and matheuristic, *Transportation Research Part E: Logistics and Transportation Review*, 127 : 284–307, 2019.

[4] Sheng Liu, Zuo-Jun Max Shen, Xiang Ji. Urban Bike Lane Planning with Bike Trajectories: Models, Algorithms, and a Real-World Case Study. *Manufacturing and Service Operations Management* 24(5):2500-2515, 2021.

[5] Álvarez-Miranda, E., Luipersbeck, M., Sinnl, M. (2016). Optimal Upgrading Schemes for Effective Shortest Paths in Networks. In: Quimper, CG. (eds) Integration of AI and OR Techniques in Constraint Programming. CPAIOR 2016. Lecture Notes in Computer Science(), vol 9676. Springer, Cham. https://doi.org/10.1007/978-3-319-33954-2-29

[6] Juan P. Ospina, Juan C. Duque, Veronica Botero-Fernandez, Alejandro Montoya. The maximal covering bicycle network design problem, *Transportation Research Part A: Policy and Practice*, 159: 222–236, 2022.

[7] Natera Orozco Luis Guillermo, Battiston Federico, Iñiguez Gerardo and Szell Michael, 2020. Data-driven strategies for optimal bicycle network growth. R. Soc. open sci.7201130201130 http://doi.org/10.1098/rsos.201130