A routing problem raised by self-service bicycle sharing systems

Frédéric Meunier – CERMICS, Ecole des Ponts, Paris, France

Talk partially based on join works with Daniel Chemla and Roberto Wolfler Calvo.

Bikes repositioning

- Essential task when operating self-service bike sharing systems (like Bixi): repositioning of the bikes at the end of the night.
- Morning rush.
- City divided into zones: a zone = a truck.

Assumption

- The bikes do not move.
- Current allocation: x_v bikes on each station v.
- Target: y_v bikes on each station v.

$\rightarrow$ Bring the system at the target state with a truck

Formalization: routing problem on graph

Input : graph $G = (V, E)$;
$\boldsymbol{d} \in \mathbb{R}_+^E$ a distance;
$\boldsymbol{x} \in \mathbb{Z}_+^V$ initial allocation;
$\boldsymbol{y} \in \mathbb{Z}_+^V$ target allocation; with $\sum_{v \in V} x_v = \sum_{v \in V} y_v$
truck capacity K.

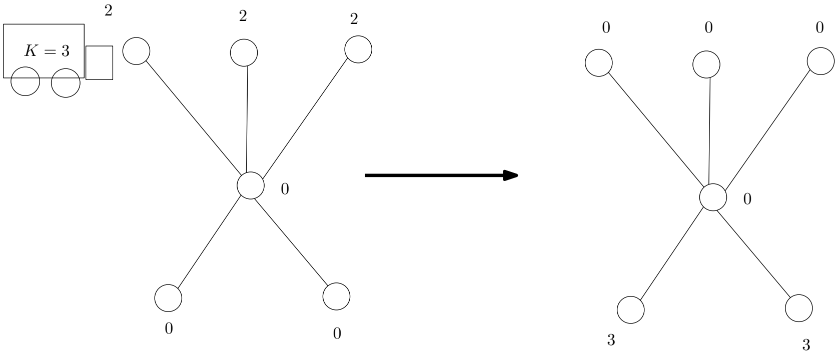
Task Find the sequence of visited stations and the bike displacements bringing the system from the state $\boldsymbol{x}$ to the state $\boldsymbol{y}$.

Objective Minimize traveled distance.

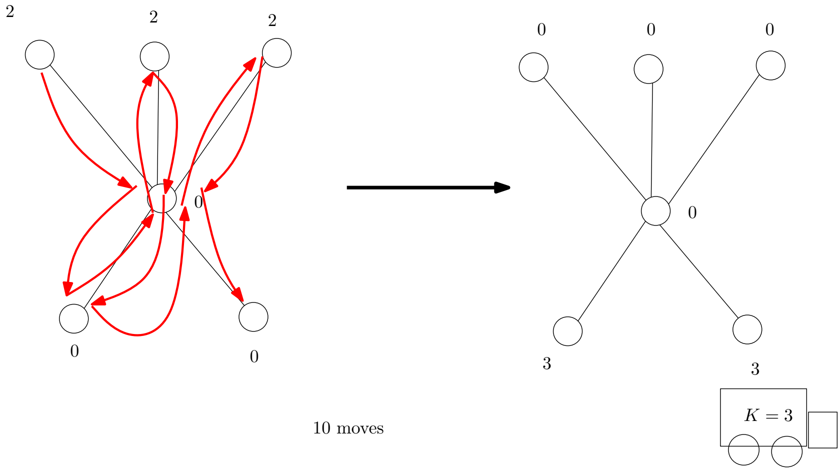
Bikes are allowed to be unloaded, and reloaded later

Preemption is allowed.

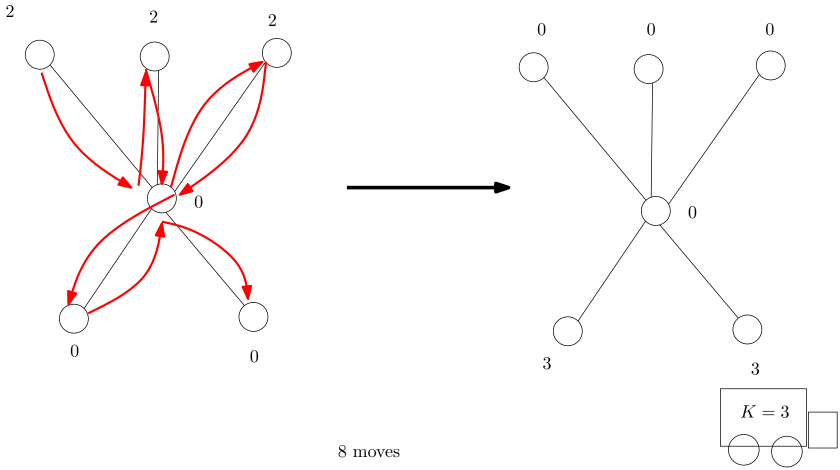
An example



If it were not allowed



Allowed: it's better



8 moves

It is an hard problem

- NP-hard problem, even if truck capacity $K = 1$
- special cases: TSP, bipartite TSP, 2-partition, split delivery, ...

Similar problems have already been studied

1-PDTSP – one-commodity pickup and delivery problem –, almost our problem, but requires Hamiltonian cycle. [Hernandez-Pérez and Salazar-Gonzáles, 2004]

Swapping Problem, almost our problem, but requires all supplies and demands to be unitary ($x_v, y_v \in \{0, 1\}$), (and several types of commodities allowed). [Annily and Hassin (1992)]

Questions that will be addressed

Practical question

- How to solve practical instances ?

Theoretical questions

- Approximation algorithms ?
- Polynomial cases ?

How to solve practical instances ?

A combinatorial encoding of the optimal solutions

[Chemla, M., Wolfler 2012]
Polynomial algorithm that finds the best loading and unloading operations for a given sequence of vertices visited by the truck.

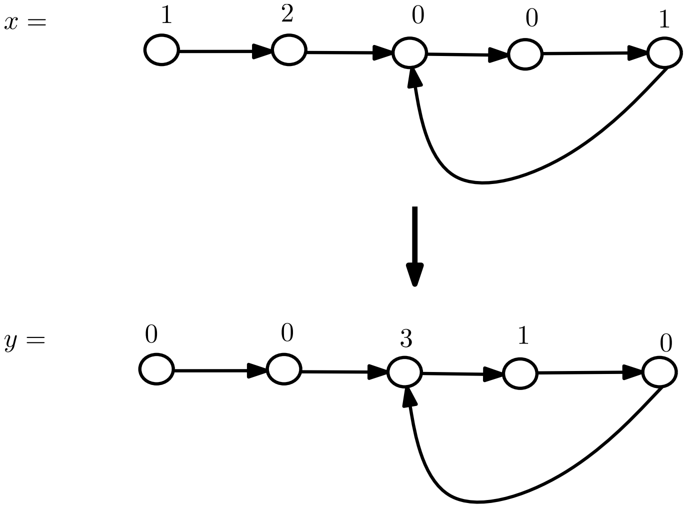
Best loading and unloading operations :

$\boldsymbol{x}'$ and $\boldsymbol{y}'$ such that

- $x'(V) = y'(V)$
- $x'_v \leq x_v$ and $y'_v \leq y_v$ for all $v \in V$
- maximizing $x'(V)$.

it solves also

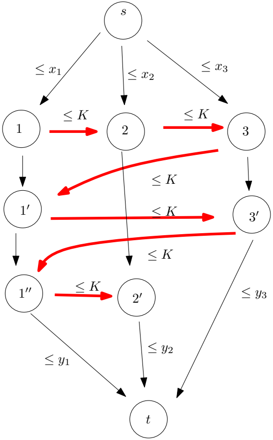
- $x'(V) = y'(V)$
- $x'_v = x_v$ for all $v \in V$
- minimizing $\sum_{v \in V} |y_v - y'_v|$.



$x =$
1
2
0
0
1
$y =$
0
0
3
1
0

The polynomial encoding is possible via max flow

Stations: $1, 2, 3$.
Sequence: $1 \to 2 \to 3 \to 1 \to 3 \to 1 \to 2$



A local search

We can limit the exploration to sequences of vertices, regardless of the number of bikes carried by the truck.

Local changes

- 2-OPT
- vertex deletion
- vertex addition
- ...

Iterating local changes $\rightarrow$ local search

Note that the local search is able to deal with non-feasible solutions.

A lower bound via linear optimization

$$\min \sum_{u,v \in V} d_{uv} z_{uv}$$

$$s.c. \quad \sum_{u \in V} z_{uv} = \sum_{w \in V} z_{vw} \qquad v \in V$$

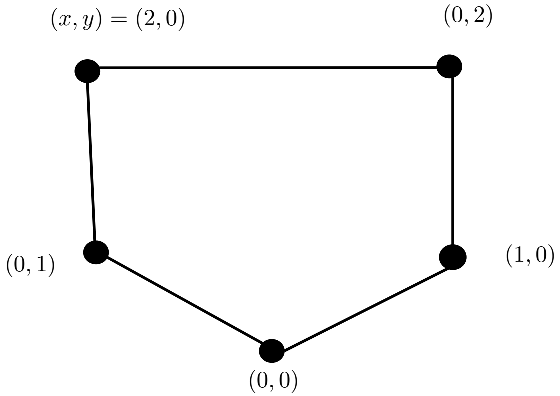
$$\sum_{u \in X,\, v \notin X} z_{uv} \geq \left\lceil \frac{|x(X) - y(X)|}{K} \right\rceil \qquad X \subseteq V \setminus \{0\}$$

$$z_{uv} \in \mathbb{Z}_+ \qquad u, v \in V$$

Solved by branch-and-cut.

It is only a lower bound

$K = 2$

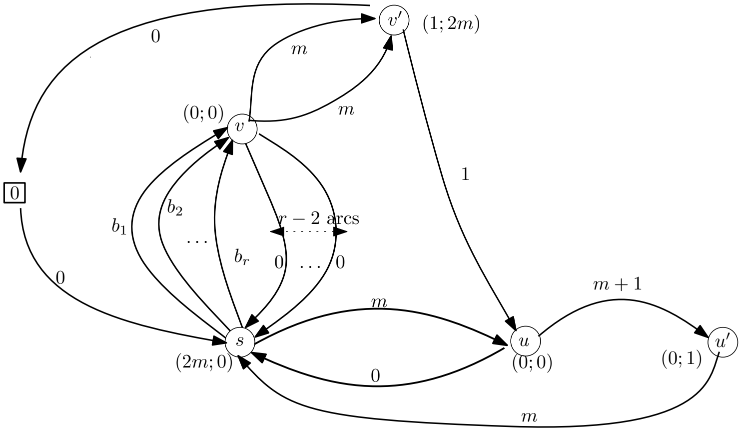


which can be optimal without being able to check it

[Chemla, M., Wolfler 2012]
Deciding whether a feasible solution of the linear program is a feasible solution for our problem is NP-complete.

If the solution is given by the number of times each edge is used, we cannot check in polynomial time whether it is a feasible solution.

since it contains 2-partition as a special case



Whole algorithm: local search initialized by branch-and-cut

- compute a (non-necessary feasible) sequence by solving the linear program (branch-and-cut) transformed into an Eulerian circuit

- apply tabu search

PC AMD Athlon 5600+ clocked at 2.8 GHz, with 16 MB RAM. CPLEX for the linear program.

Computational results for local search and branch-and-cut

Instance	n	K	UB	Time	LB	Gap %
n20A	20	10	4702	7	4702.00	0.00
n20C	20	10	6013	14	6012.00	0.02
n20B	20	10	4769	8	4769.00	0.00
n20A	20	30	3583	4	3583.00	0.00
n20E	20	30	4556	5	4299.00	5.98
n20F	20	30	4108	5	4108.00	0.00
n40E	40	10	6424	2253	6424.00	0.00
n40F	40	10	7095	10509	6760.00	4.96
n40J	40	10	6268	10067	6267.00	0.02
n40A	40	30	4949	178	4949.00	0.00
n40C	40	30	4692	450	4644.00	1.03
n40B	40	30	5110	301	5110.00	0.00
n60H	60	10	8208	11328	7707.44	6.49
n60B	60	10	8723	11312	7508.53	16.17
n60A	60	10	8010	11349	7276.80	10.08
n60G	60	30	6360	1264	6360.00	0.00
n60I	60	30	6766	8234	6390.00	5.88
n60H	60	30	6081	1835	5992.00	1.49

Results for instances with a mean of 10 bikes per station.

Computational results for local search and branch-and-cut

Instance	n	K	UB	Time	LB	Gap %
n20B	20	10	9883	71	9883.00	0.00
n20C	20	10	14040	137	14039.00	0.01
n20D	20	10	14925	247	14925.00	0.00
n20B	20	30	4769	16	4769.00	0.00
n20C	20	30	6013	23	6012.00	0.02
n20D	20	30	5989	16	5989.00	0.00
n40E	40	10	13159	1786	13159.00	0.00
n40F	40	10	15410	11309	14456.90	6.59
n40I	40	10	14849	2531	14849.00	0.00
n40E	40	30	6424	1024	6424.00	0.00
n40F	40	30	7240	10239	6571.83	10.17
n40I	40	30	6901	2144	6901.00	0.00
n60F	60	10	17696	11414	16925.71	4.55
n60A	60	10	18755	11075	15789.56	18.78
n60J	60	10	17462	11136	15774.62	10.70
n60H	60	30	8120	11334	7608.96	6.72
n60C	60	30	9818	11227	8313.06	18.10
n60J	60	30	8407	11357	7642.33	10.01

Results for instances with a mean of 30 bikes per station.

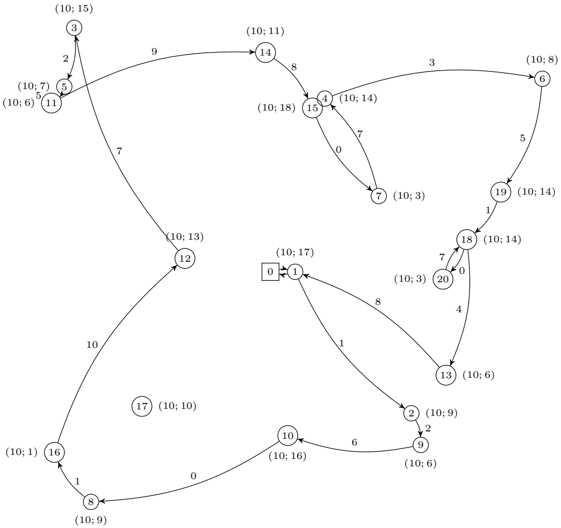


Figure: An optimal solution for an instance with $n = 20$, $K = 10$ et $\frac{1}{20}x(V) = 10$

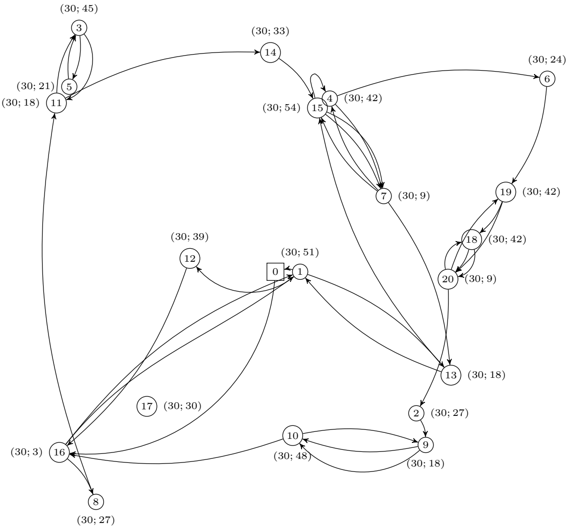


Figure: An optimal solution for an instance with $n = 20$, $K = 10$ et $\frac{1}{20}x(V) = 30$

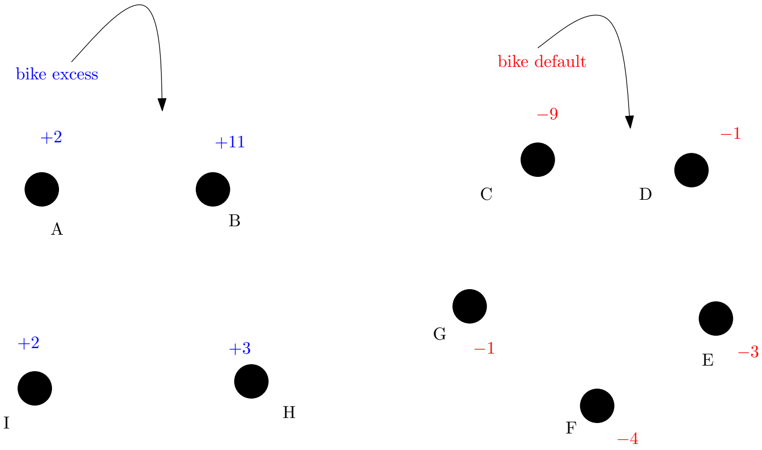
Approximation algorithm ?

A 9.5-approximation algorithm

[M. et al., 2011]
There is a 9.5-approximation algorithm.

Generalization of Chalasani-Motwani algorithm for the Swapping Problem with only one type of objects
(*i.e.* our problem with $x_v + y_v \leq 1$ for all v).

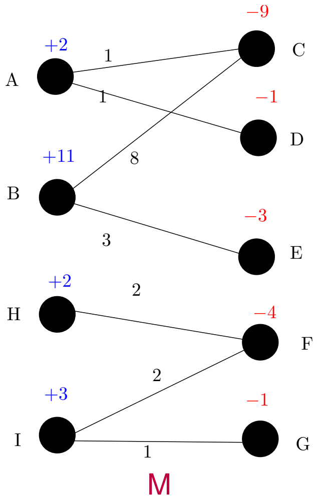
An example of input



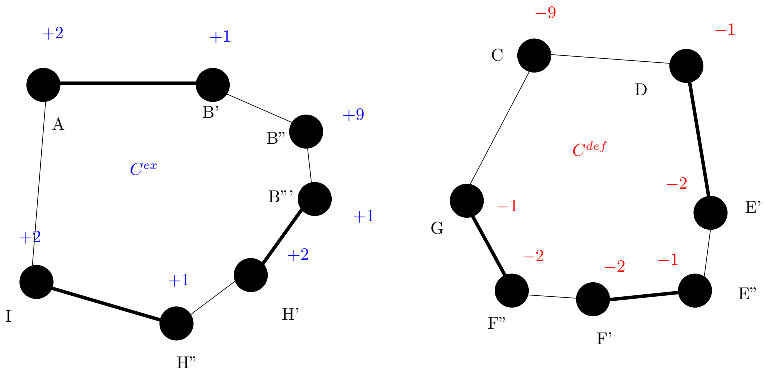
The steps of the algorithm are...

- Perfect "b-matching" M of minimal cost between excess vertices and default vertices
- Tour C^{ex} passing through all excess vertices
- Tour C^{def} passing through all default vertices
- Split these tours in subpaths with excess or default a multiple of K bikes
- Transfer bikes via M from excess subpaths to default subpaths

Perfect b-matching between excess vertices and default vertices

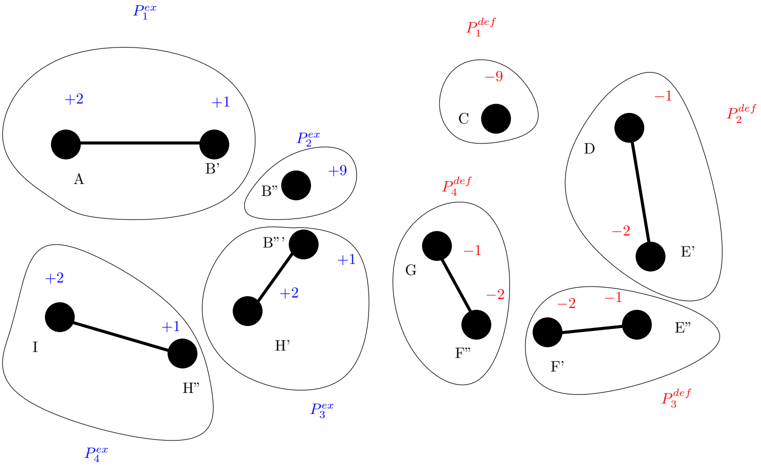


Tour on the excess vertices and tour on the default vertices



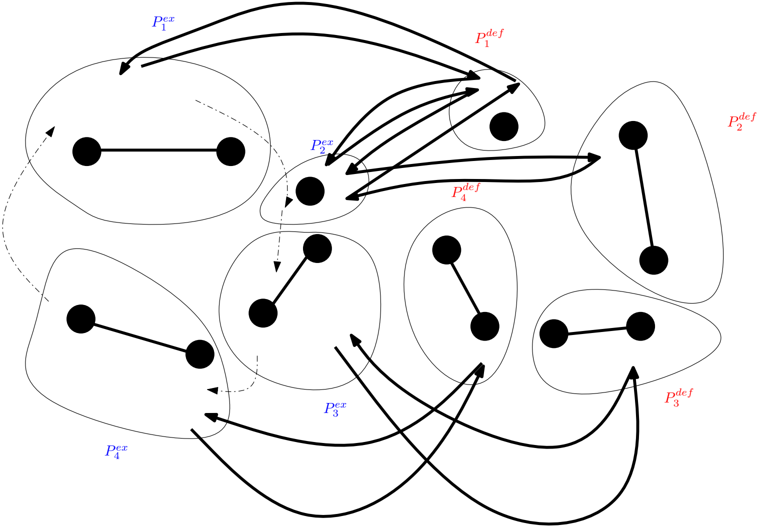
$K = 3$

Subpaths of multiple K bikes



$K = 3$

Everything put together



We obtain a 9.5-approximation algorithm

$$
\begin{aligned}
SOL &\leq 2C^{ex} + 2C^{def} + 2/K\,Mo + C^{ex} \\
&\leq 4.5OPT + 3OPT + 2OPT \\
&= 9.5OPT
\end{aligned}
$$

via Christofidès heuristics and König's theorem (colouring version).

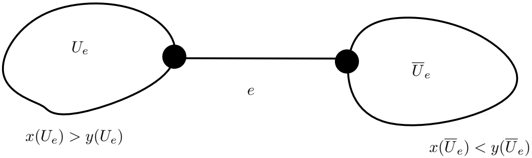
Polynomial cases ?

A polynomial case: tree

[M. et al., 2011]
Polynomial time solvable if G is a tree.

In addition, for each $e \in E$

$$\text{truck uses edge } e \simeq 2\left\lceil \frac{x(U_e) - y(U_e)}{K} \right\rceil$$



A greedy algorithm

If there are stations which have not reached their target state, repeat

1. compute $Q_1, Q_2, \dots, Q_s$ connected components of $G \setminus \{v\}$

   v current position of the truck

2. <u>If there is a Q_i with bikes in excess</u>
   - choose such a Q_i,
   - unload all bikes of the truck,
   - enter Q_i.
3. <u>Otherwise</u>
   - choose a Q_i with an *unbalanced* vertex
   - load bikes from v till the truck carries $\min(K, y(Q_i) - \tilde{x}(Q_i))$ bikes,
   - enter Q_i.

(in case of a tie, choose Q_i without depot)

Other polynomial case ?
An exact algorithm ?
Polynomial encoding of solutions ?

Open question 1. Polynomially solvable if G is a cycle ?

Open question 2. Existence of an efficient exact algorithm ?

Open question 3. Polynomial encoding of optimal solutions ?

- $K = 1$, two stations u and v, n bikes on u, all bikes have to be carried from u to v.
- size of the input: $\simeq \log_2(n)$
- optimal solution = $uvuvuvuv\ldots$ of length $2n$.

Thank you