Dynamic programming for the Electric Vehicle Orienteering Problem with multiple technologies

Odysseus 2018, Cagliari

Dario Bezzi, Alberto Ceselli, Giovanni Righini

Department of Computer Science,
University of Milan, Italy

June 4^{th}, 2018

Motivation: the EVRP

The Electric VRP is usually defined with the following data:

- a complete graph $G = (V, R, E)$, where
  - V is the set of customer vertices;
  - R is the set of recharge stations;
  - E is the set of edges (weighted, non oriented);
- a special vertex 0 in R (the depot);
- for each edge $e \in E$:
  - an amount of energy consumed d_e;
  - a traveling time t_e;
- a demand q_i for each $i \in V$;
- a service time s_i for each vertex $i \in V \cup R$;
- the number of available vehicles, W;
- the vehicle capacity Q;
- the maximum route duration T;
- the battery capacity B.

The EVRP with multiple technologies: references

We consider the variation of the EVRP with

- battery amortization (easy to handle);
- partial recharges (makes the problem mixed-integer);
- multiple technologies (makes the problem considerably harder).

EVRP problem:

- NP-hard
- heuristics (Schneider et al., TS 2014, Felipe et al., TR-E 2014)
- exact [for single technology] (Desaulniers et al., OR 2016, Andelmin and Bartolini, TS 2017)

The EVRP with multiple technologies: data

Additional data:

- a fixed cost f to be paid for each recharge operation;
- a set H of recharge technologies;
- a technology $h_i \in H$ for each $i \in R$ (R_h is the set of stations equipped with technology $h \in H$);
- for each technology $h \in H$:
  - a unit recharge time ρ_h;
  - a unit recharge cost γ_h;

Assumption 1: $\rho_{h'} > \rho_{h''} \wedge \gamma_{h'} < \gamma_{h''} \qquad \forall h' < h'' \in H.$
Assumption 2: a technology $h = 0$ exists at the depot, with $\rho_0 = 0$ and $\gamma_0 = 0$.

The EVOP: problem definition

When designing a column generation algorithm to be implemented within a branch-and-price for the EVRP, the Electric Vehicle Orienteering Problem arises as the pricing sub-problem:

- Single vehicle;
- Prizes $\beta \geq 0$ on customer vertices (dual variables): negative cost arcs and cycles;
- Penalty $\eta \geq 0$ for the use of the vehicle (dual variable);
- Elementarity constraints on customer vertices, but stations can be visited multiple times;
- Empty legs (direct edges between stations) can be traversed multiple times;
- A minimum cost recharge plan must be decided, for any given route.

Pricing is commonly done with dynamic programming.

The state

The D.P. state $\mathcal{L} = (u, S, \phi, \tau, c, \mathcal{P})$ corresponding to a path from the depot to vertex u is defined as follows:

- $u \in N \cup R$ is the last vertex reached by the path, that starts from the depot;
- $S \subseteq N$ is the set of customers visited by the path;
- ϕ is the amount of capacity already consumed: $\phi = \sum_{i \in S} q_i$;
- τ is the time needed to reach the *reference point* after traveling along the path;
- c is the (hopefully negative) cost to reach the *reference point* after traveling along the path;

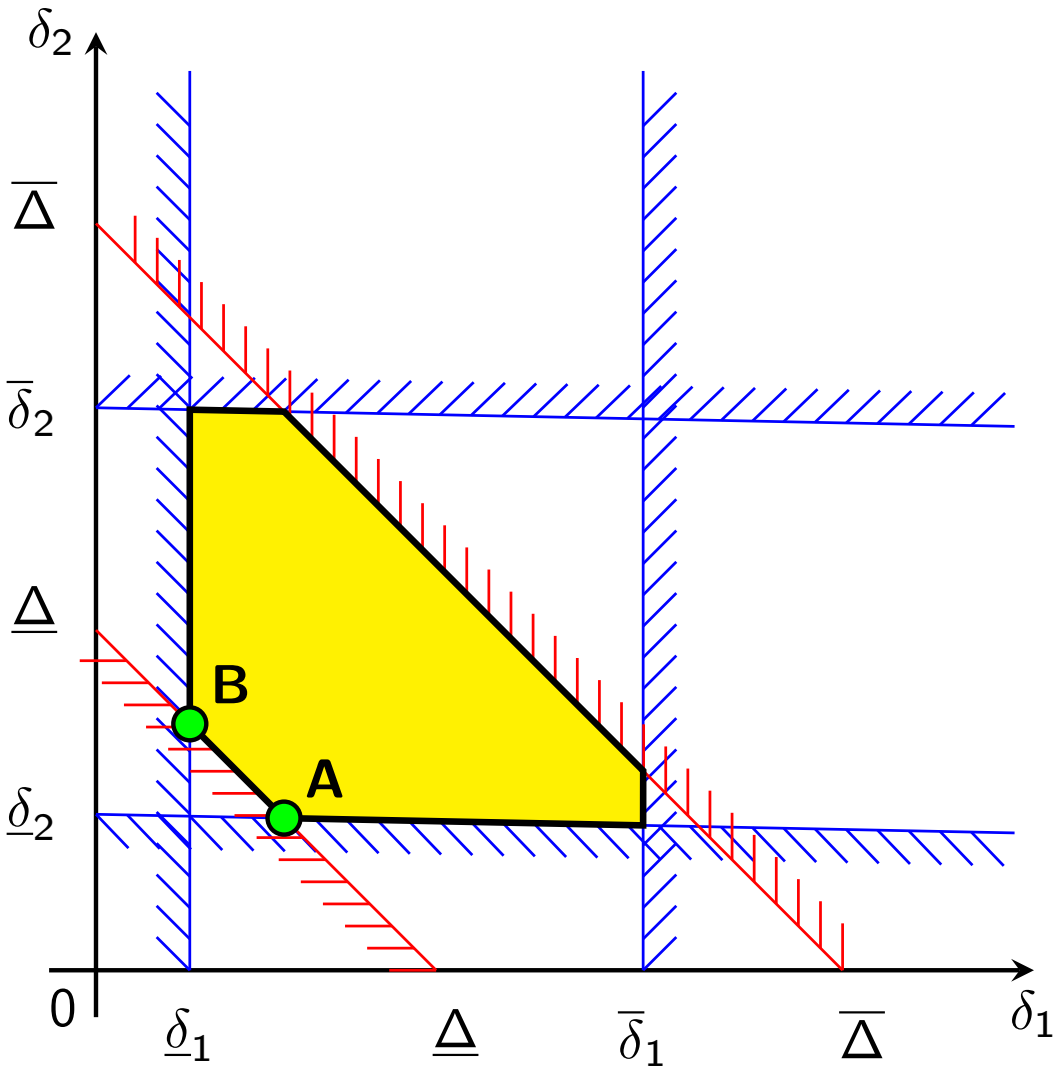
The feasible recharge polyhedron

The FRP $\mathcal{P} = (\underline{\Delta}, \overline{\Delta}, \underline{\delta}, \overline{\delta})$ is defined in as many dimensions as the number of technologies and it has a special structure:

- vectors $\underline{\delta} = \{\underline{\delta}_0, \ldots \underline{\delta}_H\}$ and $\overline{\delta} = \{\overline{\delta}_0, \ldots \overline{\delta}_{\mathcal{H}}\}$ represent the minimum and maximum amounts of energy δ_h that can be recharged for each $h \in \mathcal{H}$;
- the two scalars $\underline{\Delta}$ and $\overline{\Delta}$ represent the minimum and maximum total amount of recharged energy along the path;
- the vector $\{\hat{\delta}_0, \ldots, \hat{\delta}_H\}$ represents the coordinates of a special vertex of the FRP: the *reference point*.

The *reference point* is the minimum cost vertex of the FRP.

The feasible recharge polyhedron



The reference point of a feasible recharge polyhedron $\mathcal{P}$ is the vertex of $\mathcal{P}$ of minimum cost (point **A** in the figure).

Figure: A FRP in two dimensions (we assume $\gamma_1 < \gamma_2$).

Initialization

Initialization. The initial state is $\{0, \emptyset, 0, 0, 0, \mathcal{P}^0\}$.

The polyhedron $\mathcal{P}^0$ is defined by:

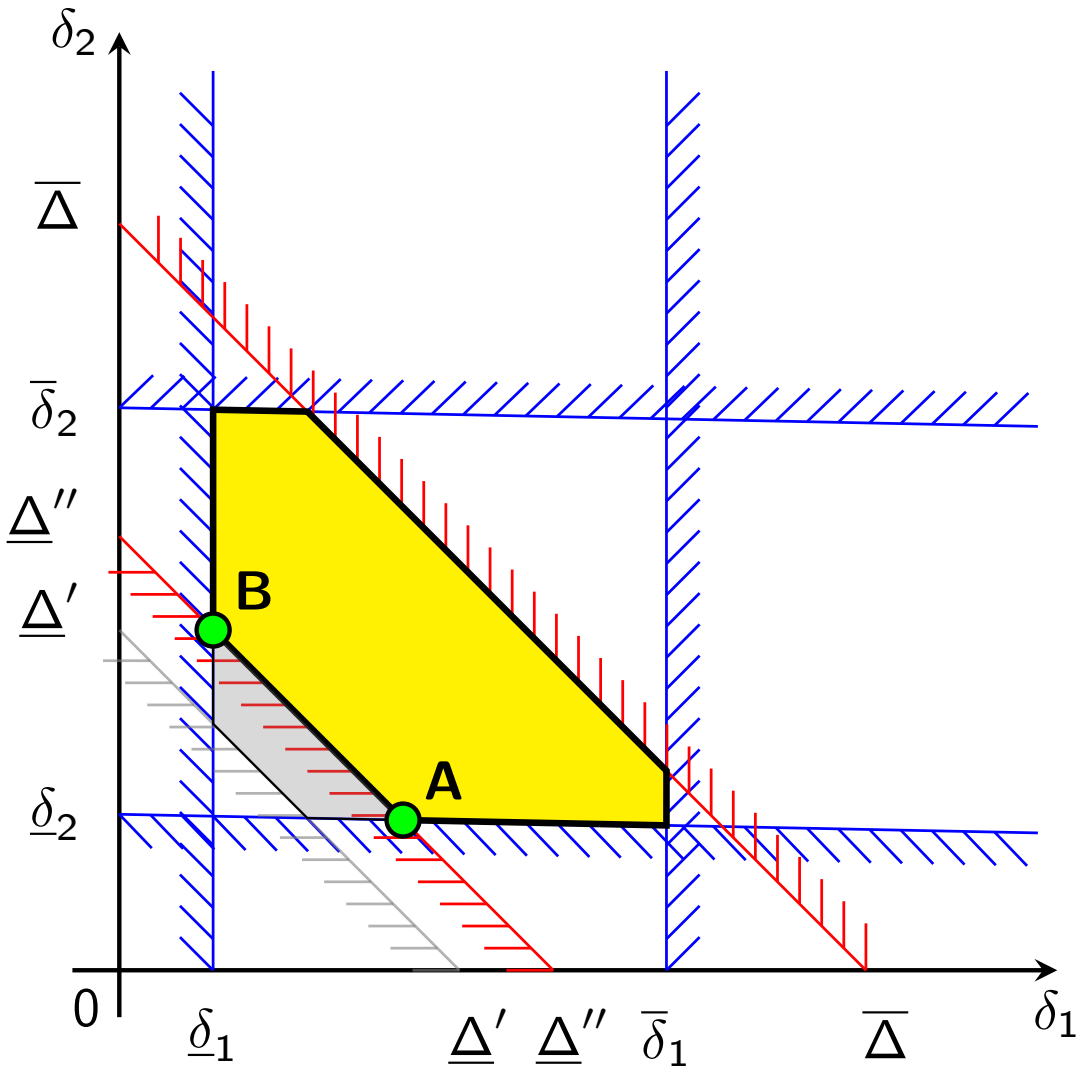
- $\underline{\delta_h} = 0 \ \forall h \in H$
- $\overline{\delta_h} = 0 \ \forall h \in H \backslash \{0\}$
- $\overline{\delta_0} = B$
- $\underline{\Delta} = 0$
- $\overline{\Delta} = B$

Extension along an edge

When a state $\mathcal{L}'$ of vertex i is extended to a state $\mathcal{L}''$ of vertex j along an edge e the extension rules are:

- $S'' = S' \cup \{j\}$
- $\phi'' = \phi' + q_i/2 + q_j/2$

Extension along an edge



When the vehicle travels along an edge, $\underline{\Delta}$ (and possibly $\underline{\delta}_h$ for some h) increases, restricting $\mathcal{P}$.

Figure: Restriction of the FRP when an edge is traversed

Reference point update

1: $\underline{\Delta}'' \leftarrow \underline{\Delta}' + d_e$
2: $Z \leftarrow d_e$
3: **for** $h = 0, \ldots, H$ **do**
4: $\quad \epsilon \leftarrow \min\{Z, \overline{\delta}'_h - \hat{\delta}'_h\}$
5: $\quad \hat{\delta}''_h \leftarrow \hat{\delta}'_h + \epsilon$
6: $\quad Z \leftarrow Z - \epsilon$
7: **end for**

The new bounds are:

- $\overline{\delta}''_h = \overline{\delta}'_h \quad \forall h \in H$
- $\underline{\delta}''_h = \max\{\underline{\delta}'_h, \underline{\Delta}'' - \sum_{k \neq h} \overline{\delta}''_k\} \quad \forall h \in H$

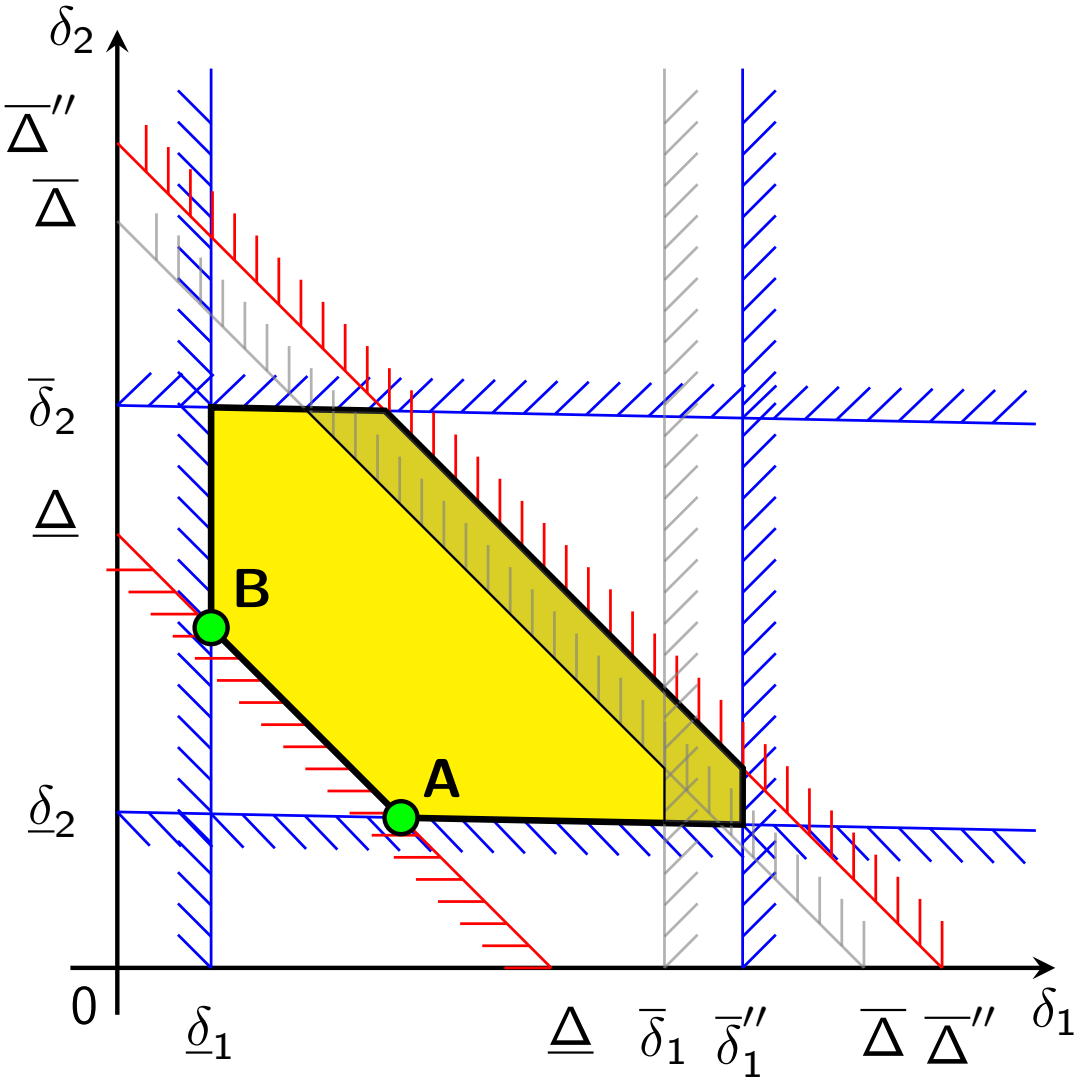
Extension along an edge

The extensions for time and cost depend on the new coordinates $\hat{\delta}''$ of the reference point:

- $\tau'' = \tau' + s_i/2 + t_e + s_j/2 + \sum_{h \in H} \rho_h(\hat{\delta}''_h - \hat{\delta}'_h)$
- $c'' = c' - \beta_i/2 - \beta''/2 + \sum_{h \in H} \gamma_h(\hat{\delta}''_h - \hat{\delta}'_h)$

The prize β for station vertices is set to $-\eta$ for the depot and to $-f$ for the other stations.

Extension in a station



When the vehicle visits a station equipped with technology h, $\overline{\Delta}$ and $\overline{\delta}_h$ increase, enlarging $\mathcal{P}$.

Figure: Extension of the FRP when a station is visited.

Extension in a station

When a state $\mathcal{L}'$ extended to a state $\mathcal{L}''$ in a station with technology k the extension rules are:

- $S'' = S'$
- $\phi'' = \phi'$
- $\tau'' = \tau'$
- $c'' = c'$
- $\hat{\delta}'' = \hat{\delta}'$
- $\underline{\Delta}'' = \underline{\Delta}'$
- $\overline{\Delta}'' = \underline{\Delta}' + B$
- $\underline{\delta}'' = \underline{\delta}'$
- $\overline{\delta}''_h = \overline{\delta}'_h \ \forall h \in H, h \neq k$
- $\overline{\delta}''_k = \overline{\delta}'_k + (\overline{\Delta}'' - \overline{\Delta}')$

Feasibility check

For an extension from a state $\mathcal{L}' = \{i, S', \phi', \tau', c', \mathcal{P}'\}$ of vertex i to a state of vertex j along an edge $e = [i, j]$ to be feasible, the following conditions are tested ($q_i = 0$ if $i \in R$):

- Elementarity: $j \notin S'$
- Capacity: $\phi' + q_i/2 + q_j/2 \leq Q$
- Duration: $\tau' + s_i/2 + t_e + s_j/2 + b_j \leq T$
- Energy: $\overline{\Delta}' - \underline{\Delta}' \geq d_e + m_j$

Not including /2 causes bugs!.
where b_j and m_j are precomputed for each vertex j:

- b_j is the travelling time of the direct edge from j to the depot;
- m_j is the minimum amount of energy needed to reach a station (the nearest one) from j.

Forbidden extensions

The extension to a station r is forbidden if no customer vertex has been visited after the last visit to r. Keeping a boolean flag for each station is enough for this test.

The direct extension from a station B to a station C after a direct extension from a station A to the station B is forbidden if the direct extension from A to C is possible and B and C have the same technology.

Dominance

A state $\mathcal{L}' = \{i, S', \phi', \tau', c', \mathcal{P}'\}$ dominates a state $\mathcal{L}'' = \{i, S'', \phi'', \tau'', c'', \mathcal{P}''\}$ only if:

- $S' \subseteq S''$
- $\phi' \leq \phi''$
- $\tau' \leq \tau''$
- $c' \leq c''$
- $\mathcal{P}' \supseteq \mathcal{P}''$ when the two reference points are made coincident.

Remark. The test $S' \subseteq S''$ is implemented by bitwise comparison and takes $O(1)$.

Dominance between FRP

The test whether $\mathcal{P}' \supseteq \mathcal{P}''$ when the two reference points are made coincident is the following:

- $\overline{\Delta}' - \underline{\Delta}' \geq \overline{\Delta}'' - \underline{\Delta}''$
- $\overline{\delta}'_h - \hat{\delta}'_h \geq \overline{\delta}''_h - \hat{\delta}''_h \quad \forall h$
- $\hat{\delta}'_h - \underline{\delta}'_h \geq \hat{\delta}''_h - \underline{\delta}''_h \quad \forall h$

Bi-directional extension rules

A common way to speed-up the D.P. algorithm is to extend the states in both directions.
This implies suitable definitions for:

- backward extension rules,
- extension stop criterion,
- rules for joining semi-routes.

In our case, we do not need backward extension rules because the graph is assumed to be symmetric. Therefore the same semi-routes can be traversed in either direction at the same (reduced) cost in the same time.

Extension stop criterion

The extension stop criterion must guarantee that every feasible route can be generated by a pair of semi-routes.

We stop the extension when half of a critical resource has been consumed.

As a critical resource we can select

- capacity q;
- time τ.

If an extension would exceed $Q/2$ in capacity consumption or $T/2$ in time consumption, then the extension is not done.

Adaptive choice of the critical resource

The best choice of the critical resource depends on the specific instance at hand: the most binding resource is the best candidate for being selected as critical.

We devised a simple self-adapting technique:

- all states are extended until either the capacity limit or the time limit is reached;
- when all extensions are over, the states where extensions have been stopped are counted;
- the critical resource is set to the one which stops most of them;
- extensions stopped by the other resource(s) are done.

Critical resource

Choosing the wrong critical resource could lead to a huge waste of time...

istance	$\|V\|$	$\|R\|$	Q	T	B	r.c.	s	labels
$A1 - N030$	30	10	2300	480	160	T	4.67	1361
$A1 - N030$	30	10	2000	480	160	T	4.64	1361
$A1 - N030$	30	10	1200	480	160	T	4.31	1299
$A1 - N030$	30	10	800	480	160	T	2.00	866
$A1 - N030$	30	10	700	480	160	Q	1327	42456
$A1 - N030$	30	10	1000	380	160	T	0.232	228
$A1 - N030$	30	10	600	380	160	Q	9.31	3403
$A1 - N030$	30	10	600	280	160	Q	0.065	136
$A1 - N030$	30	10	1200	180	160	T	0.002	4

Table: Number of labels (semi-routes) generated for the same instance with different values for Q and T

Join energy: join edge

When two semi-routes $\mathcal{L}' = (i, S', \phi', \tau', c', \mathcal{P}')$ and $\mathcal{L}'' = (j, S'', \phi'', \tau'', c'', \mathcal{P}'')$ are joined along the edge $e = [i, j]$ they give origin to a complete route in which the energy consumption is given by $\underline{\Delta}' + \underline{\Delta}'' + d_e$.

The additional cost and time needed by the energy consumption d_e to traverse edge e can be quickly lower bounded by $d_e\rho^*$ (time) and $d_e\gamma^*$ (cost), where ρ^* and γ^* are the coefficients of the most convenient technologies among those available in either semi-route (i.e. technologies for which $\hat{\delta}_h < \overline{\delta}_h$).

Join energy: tech replacement

When the route is generated from two semi-routes, traversing it implies traversing one of the two semi-routes *to* the depot, instead of *from* the depot.

The energy consumption along the reversed semi-route is the same, but the technology used cannot be the same. When semi-routes are built, they are built *from* the depot, assuming the possibility of up to B units of energy initially available from technology 0. For a reversed semi-route this is not possible.

Therefore, after joining two semi-routes, $\max\{0, \hat{\delta}'_0 + \hat{\delta}''_0 - B\}$ units of energy recharged with technology 0 must be replaced using a different technology. In the worst case (routes whose overall consumption is larger than B) this implies the replacement of B units of energy. We call this amount join energy, E^{join}.

The additional cost and time due to the join energy can be lower bounded by $E^{join}\rho^*$ (time) and $E^{join}\gamma^*$ (cost), as before.

Join: feasibility test

Consider two states $\mathcal{L}' = (i, S', \phi', \tau', c', \mathcal{P}')$ and $\mathcal{L}'' = (j, S'', \phi'', \tau'', c'', \mathcal{P}'')$ and the edge $e = [i, j]$. The following (necessary but not sufficient) conditions are tested to check whether the two semi-routes can be joined to produce a feasible route.

$$\begin{cases} S' \cap S'' = \emptyset \\ \phi' + \phi'' \leq Q - (q_i/2 + q_j/2) \\ \tau' + (E_{join} + d_e)\rho^* + \tau'' \leq T - (s_i/2 + t_e + s_j/2) \\ (\overline{\Delta}' - \underline{\Delta}') + (\overline{\Delta}'' - \underline{\Delta}'') \geq B + d_e \\ c' + (E_{join} + d_e)\gamma^* + c'' \leq \beta_i/2 + \beta_j/2 \end{cases}$$

The amounts in blue are due to the join edge e.
The amounts in red are due to the join energy.

Join rules

To determine the exact cost and the duration of the resulting route, we need to examine the FRP resulting from joining the two semi-routes. From $\mathcal{P}' = (\underline{\Delta}', \overline{\Delta}', \underline{\delta}', \overline{\delta}')$ and $\mathcal{P}'' = (\underline{\Delta}'', \overline{\Delta}'', \underline{\delta}'', \overline{\delta}'')$, we obtain $\mathcal{P}^* = (\underline{\Delta}^*, \overline{\Delta}^*, \underline{\delta}^*, \overline{\delta}^*)$ as follows:

- $\underline{\Delta}^* = \underline{\Delta}' + \underline{\Delta}''$
- $\overline{\Delta}^* = \overline{\Delta}' + \overline{\Delta}''$
- $\overline{\delta}^* = \overline{\delta}' + \overline{\delta}''$
- $\underline{\delta}^* = \underline{\delta}' + \underline{\delta}''$
- $\hat{\delta}^* = \hat{\delta}' + \hat{\delta}''$

However the maximum allowed amount of energy coming from technology 0, i.e. $\overline{\delta}_0^*$, is set to B instead of $\overline{\delta}_0' + \overline{\delta}_0'' = 2B$.

Join: optimal recharge plan

After computing $\mathcal{P}^*$, we restrict it and we update the reference point in the same way as after an extension along an edge consuming an amount of energy equal to d_e.

Then, we must enforce the constraint on the maximum recharge with technology 0, which implies an increase in time and cost for transforming E^{join} units of recharged energy from technology 0 to some other technology.

Join: optimal recharge plan

Let t and c be the initial time and the initial cost of the updated reference point:

1: $h \leftarrow 0$
2: **while** $(\hat{\delta}_0^* > B) \wedge (c < 0)$ **do**
3: $\quad h \leftarrow h + 1$
4: $\quad \epsilon \leftarrow \min\{B - \hat{\delta}_0^*, \overline{\delta}_h^* - \hat{\delta}_h^*\}$
5: $\quad \hat{\delta}_h^* \leftarrow \hat{\delta}_h^* + \epsilon$
6: $\quad \hat{\delta}_0^* \leftarrow \hat{\delta}_0^* - \epsilon$
7: $\quad t \leftarrow t + \rho_h \epsilon$
8: $\quad c \leftarrow c + \gamma_h \epsilon$
9: **end while**

When $\hat{\delta}_0^*$ (if initially larger than B) goes down to B, then a feasible recharge plan of negative cost has been found;
when c (initially negative) goes up to 0, no feasible recharge plan with negative cost exists.

Join: optimal recharge plan

If the updated reference point is not feasible, because the value of t violates the duration constraint, we must determine another point in the polyhedron, trading time for cost.

This is done by another replacement of slower recharges with faster recharges, at the expense of increasing the cost.

Finding the minimum cost point in the FRP satisfying the time constraint is a special linear programming problem that can be solved efficiently.

Join: optimal recharge plan

1: $D \leftarrow \{h \in H : \hat{\delta}_h > \underline{\delta}^*_h\}$
2: $I \leftarrow \{h \in H : \hat{\delta}_h < \overline{\delta}^*_h\}$
3: **while** $(t > T) \wedge (c < 0)$ **do**
4: Select $h \in D$ and $k \in I$ such that $\frac{\gamma_k - \gamma_h}{\rho_h - \rho_k}$ is minimum.
5: $\epsilon \leftarrow \min\{\hat{\delta}_h - \underline{\delta}^*_h, \overline{\delta}^*_k - \hat{\delta}_k, \frac{t-T}{\rho_h - \rho_k}\}$
6: $\hat{\delta}_h \leftarrow \hat{\delta}_h - \epsilon$
7: $\hat{\delta}_k \leftarrow \hat{\delta}_k + \epsilon$
8: $t \leftarrow t - (\rho_h - \rho_k)\epsilon$
9: $c \leftarrow c + (\gamma_k - \gamma_h)\epsilon$
10: **end while**

When t (if initially larger than T) goes down to T, a feasible recharge plan of negative cost has been found;
when c (initially negative) goes up to 0, no feasible recharge plan with negative cost exists.

Join: optimal recharge plan

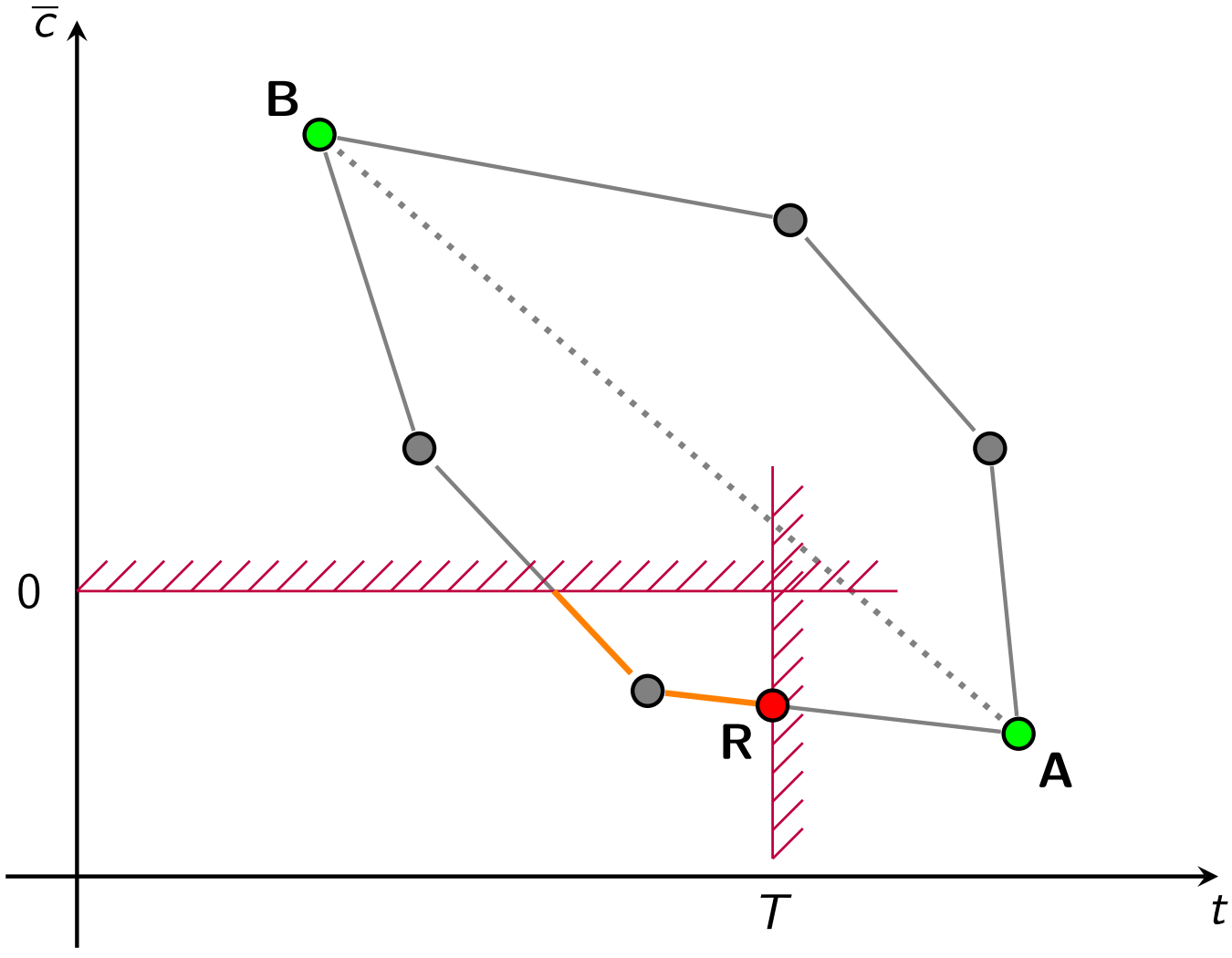


Figure: Time-cost chart with 3 techs

Join: avoiding duplicates

The same route can be generated in several different ways. This means that the join operation is executed several times, with no need. To avoid this we introduce an additional test, that depends on the selected critical resource.

Let ϕ' and ϕ'' the capacity consumptions and τ' and τ'' the time consumptions in vertices i and j, respectively. Then $\mathcal{L}'$ and $\mathcal{L}''$ are joined through edge $e = [i, j]$ only if

$$|\phi' - \phi''| \leq (q_i + q_j)/2$$

or

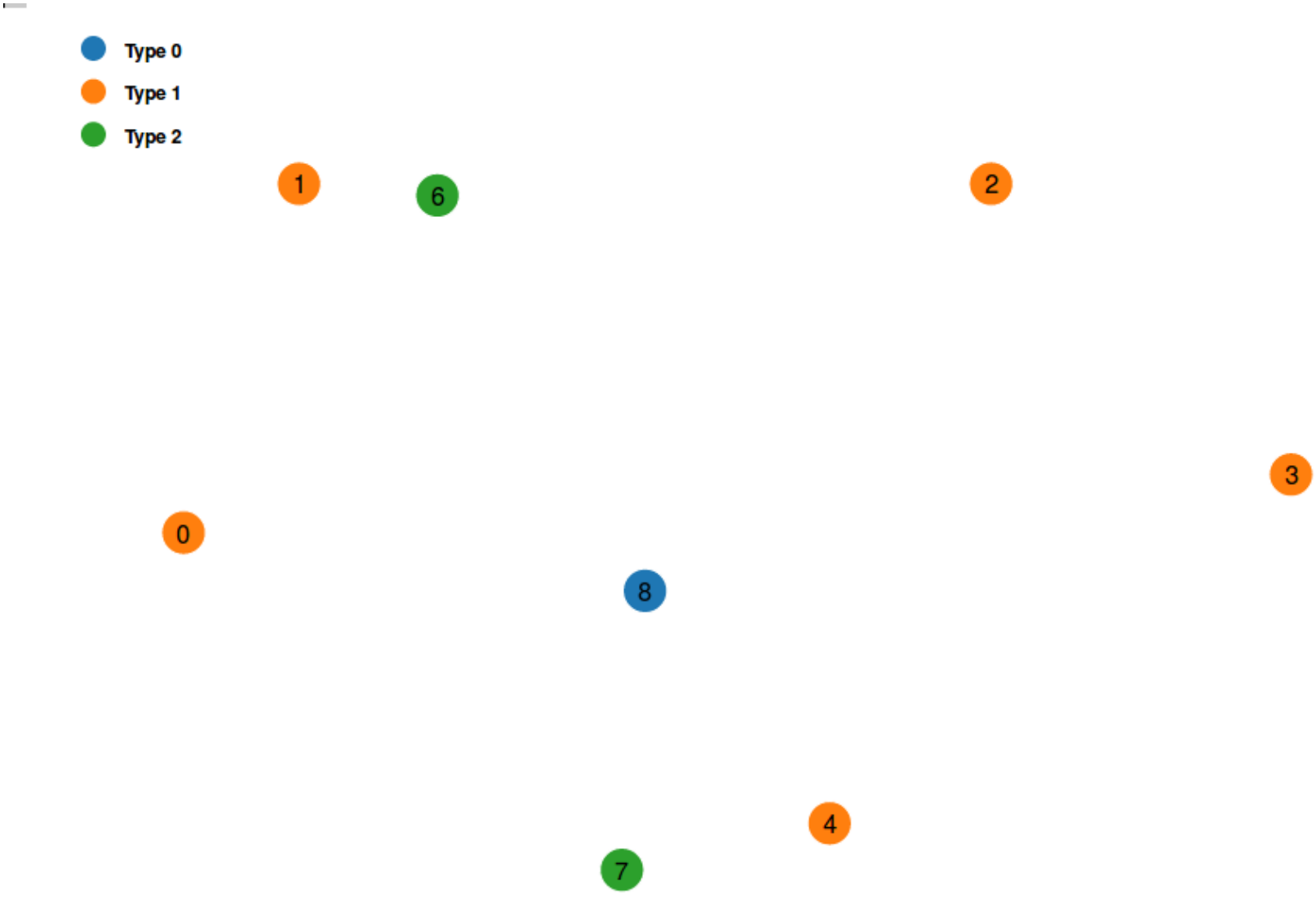
$$|\tau' - \tau''| \leq s_i/2 + t_e + s_j/2 + \rho_1(B + d_e)$$

that is, the unbalance in the selected resource consumption along the two parts of the resulting route is minimum between i and j.

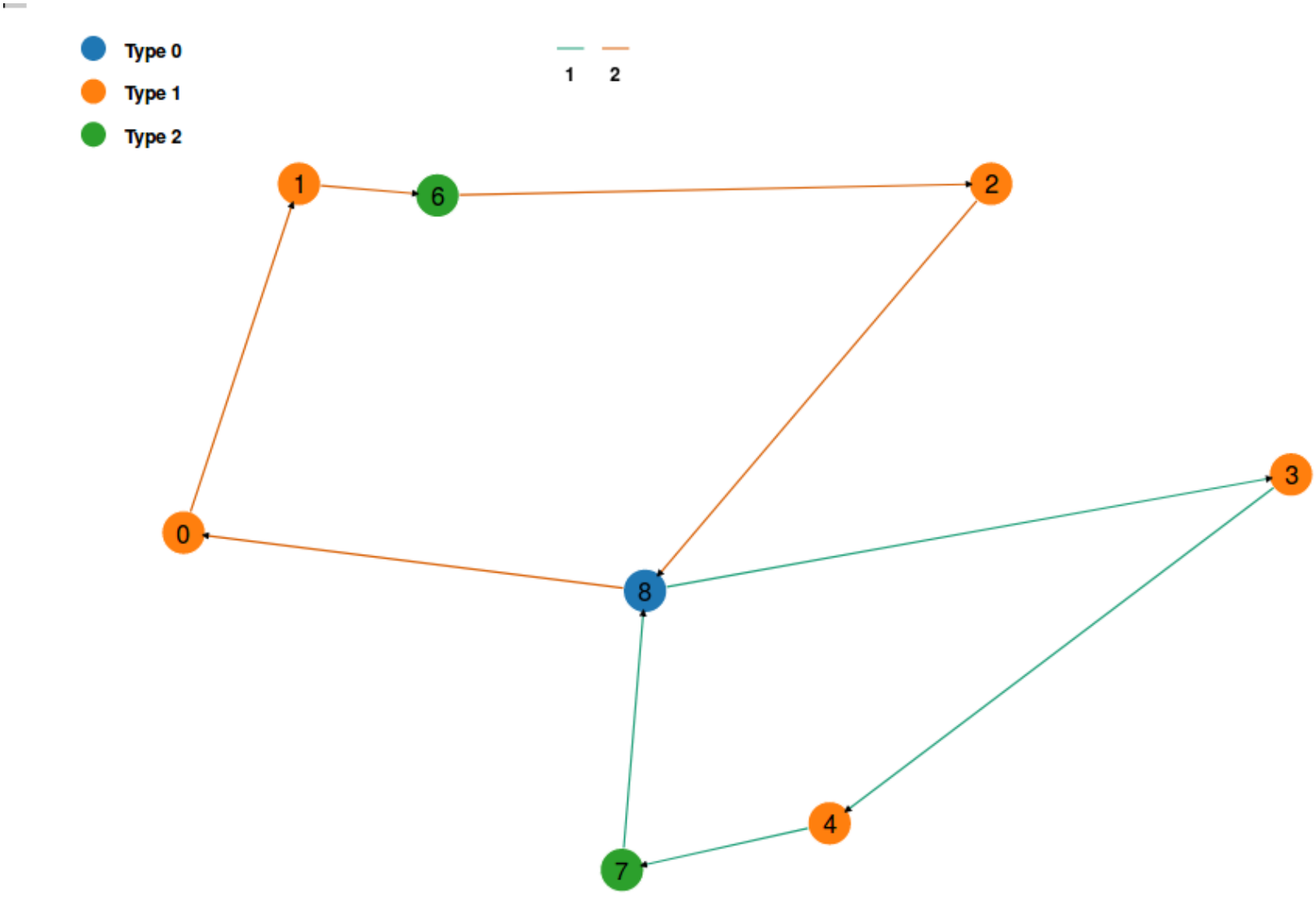
Instances

- Dedicated C++ program;
- Intel Core i3, LUbuntu LXDE environment.
- Input instances
  - Same instances used for Felipe et al., TR-E 2014;
  - Still working on VRP-rep compatibility...

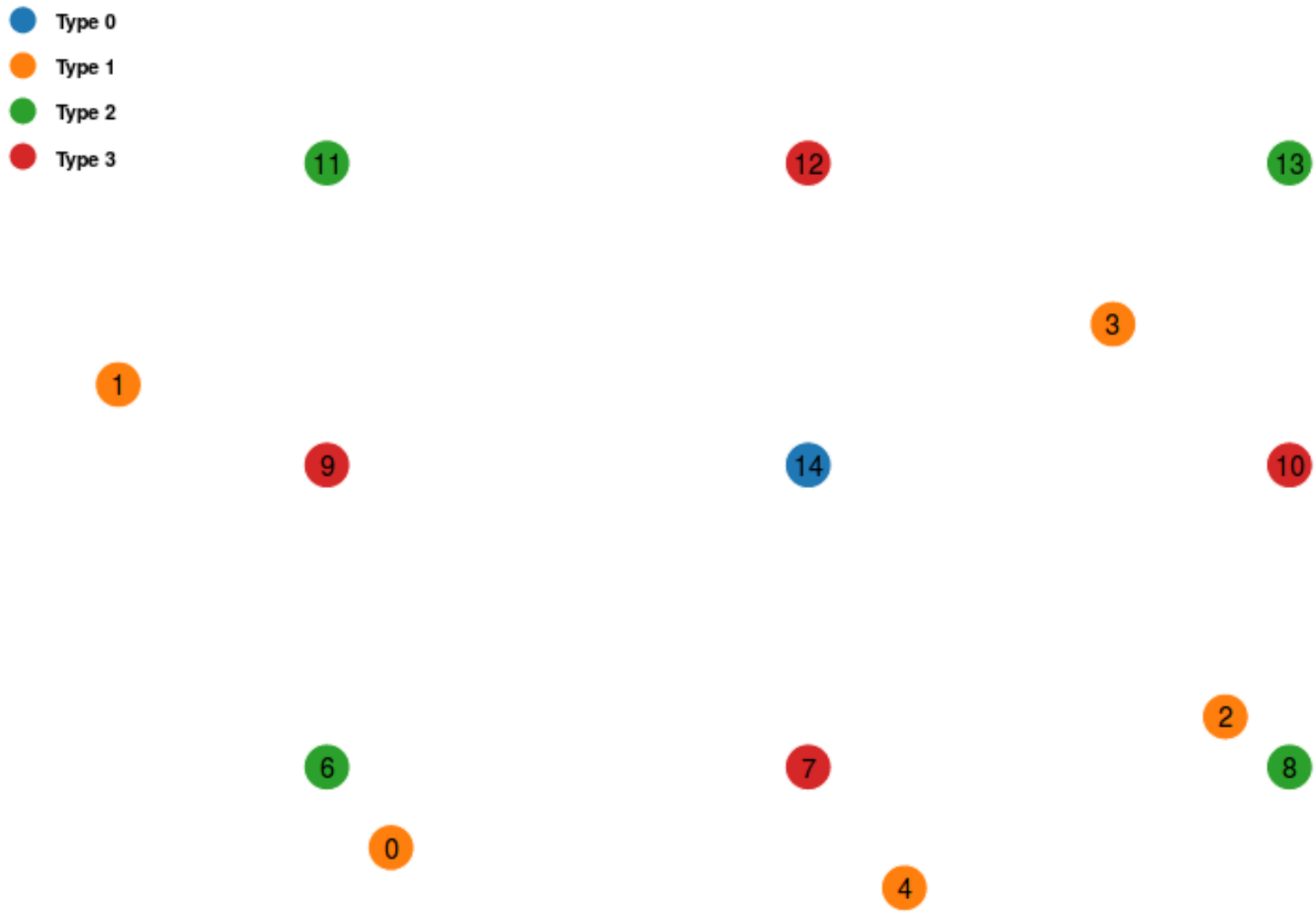
Toy instance



Toy instance



Sample instance



Results

instance	$\|V\|$	$\|R\|$	Q	T	B	r.c.	s	routes
$10 - N10$	10	10	230	480	128	T	0.294	1948
$11 - N10$	10	10	230	480	128	T	0.170	1077
$16 - N10$	10	10	230	480	128	T	1.59	10084
$17 - N10$	10	10	230	480	128	T	0.290	1775
$18 - N10$	10	10	230	480	128	T	0.333	1818
$21 - N10$	10	6	230	480	128	Q	24.0	20869
$22 - N10$	10	6	230	480	128	Q	6.77	5541
$23 - N10$	10	6	230	480	128	T	0.011	119
$27 - N10$	10	6	230	480	128	Q	1.56	484
$28 - N10$	10	6	230	480	128	Q	0.985	544
$29 - N10$	10	6	230	480	128	Q	4.5	3003

Table: Total number of feasible routes found for small instances

Results

instance	$\|V\|$	$\|R\|$	Q	T	B	r.c.	s	routes
$A0 - N030$	5	10	2300	480	160	T	0.153	801
$A1 - N030$	30	10	2300	480	160	T	30.0	100278
$A2 - N030$	30	10	2300	480	160	T	16.7	63461
$A3 - N030$	30	10	2300	480	160	T	30.1	85795
$A4 - N030$	30	10	2300	480	160	T	28.9	86083
$A5 - N030$	30	10	2300	480	160	T	118	0
$A6 - N030$	30	10	2300	480	160	T	30.0	110884
$A7 - N030$	30	10	2300	480	160	T	30.6	80641
$A8 - N030$	30	10	2300	480	160	T	27.1	106997
$A9 - N030$	30	10	2300	480	160	T	30.1	85686

Table: Number of feasible routes found in 30 seconds for larger instances

Speed-up techniques

If we are interested in multiple pricing, we do not need to generate the optimal route but rather some routes with negative reduced cost.

- Early termination, as soon as some new columns have been found.
- Sort the join edges according to their length.
- Keep the candidate join edges in a heap, to examine them according to "how much promising" they are.
- For each edge, join the routes after sorting them according to a suitably defined heuristic indicator, depending on their cost and time consumption.
- Parallel join.
- *ng*-routes, state space relaxations, etc. for elementarity constraints.

Thank you!