Exact and Heuristic Solution Methods for a VRP with Time Windows and Variable Service Start Time

Y. Arda, H. Küçükaydin, Y. Crama, S. Michelini

QuantOM - HEC - Université de Liège

VeRoLog 2015
June 10th, 2015

Table of Contents

Table of Contents

Starting scenario

- Our problem: a capacitated VRP with time windows, with additional key features:
  - Route cost depends on *total route duration*,
  - *Variable starting time* for each route,
  - Max allotted time for each route.

[1] Bettinelli, Ceselli, and Righini 2011.

[2] Desaulniers and Villeneuve 2000.

Starting scenario

- Our problem: a capacitated VRP with time windows, with additional key features:
  - Route cost depends on *total route duration*,
  - *Variable starting time* for each route,
  - Max allotted time for each route.
- Similar problems dealing with delayable departure time[1] and linear waiting costs[2] can be found in the literature.

[1] Bettinelli, Ceselli, and Righini 2011.

[2] Desaulniers and Villeneuve 2000.

Pricing subproblem for Branch-and-Price

- The pricing sub-problem that we need to solve within Branch-and-Price is an elementary shortest path problem with resource constraints (ESPPRC).

[3] Dror 1994.

[4] Developed by Feillet et al. 2004, based on Desrochers and Soumis 1988; improvements are in Righini and Salani 2008.

Pricing subproblem for Branch-and-Price

- The pricing sub-problem that we need to solve within Branch-and-Price is an elementary shortest path problem with resource constraints (ESPPRC).
- If the underlying graph may have negative cost cycles, the ESPPRC is strongly NP-Hard[3].

[3] Dror 1994.

[4] Developed by Feillet et al. 2004, based on Desrochers and Soumis 1988; improvements are in Righini and Salani 2008.

Pricing subproblem for Branch-and-Price

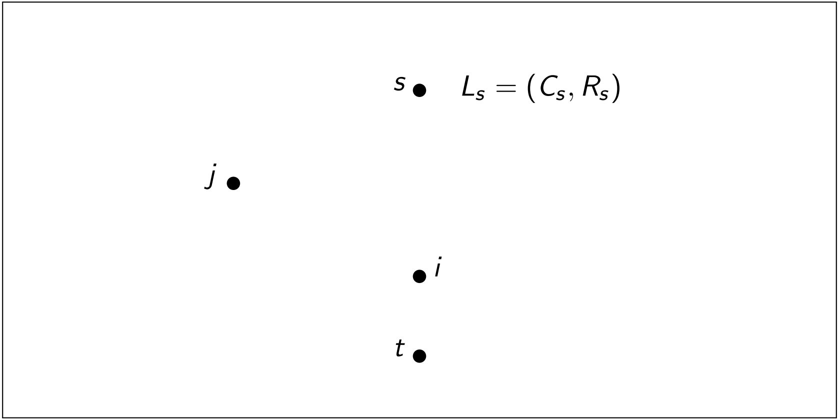
- The pricing sub-problem that we need to solve within Branch-and-Price is an elementary shortest path problem with resource constraints (ESPPRC).
- If the underlying graph may have negative cost cycles, the ESPPRC is strongly NP-Hard[3].
- The ESPPRC can be solved exactly with dynamic programming [4].

[3] Dror 1994.

[4] Developed by Feillet et al. 2004, based on Desrochers and Soumis 1988; improvements are in Righini and Salani 2008.

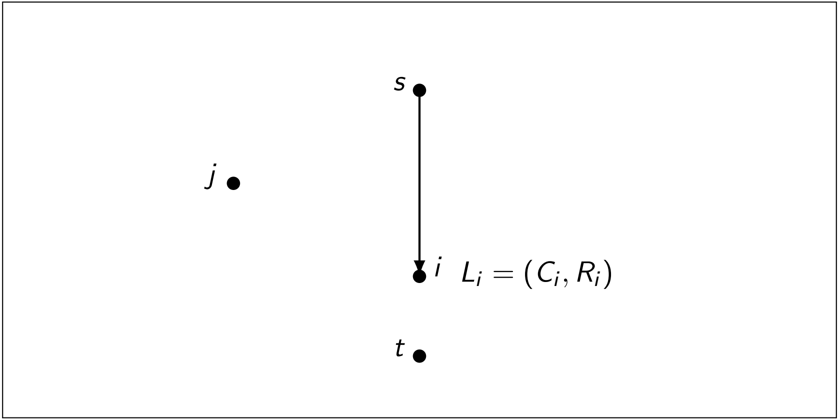
Exact Dynamic Programming for the ESPPRC

- We initialize the labeling algorithm at the source:



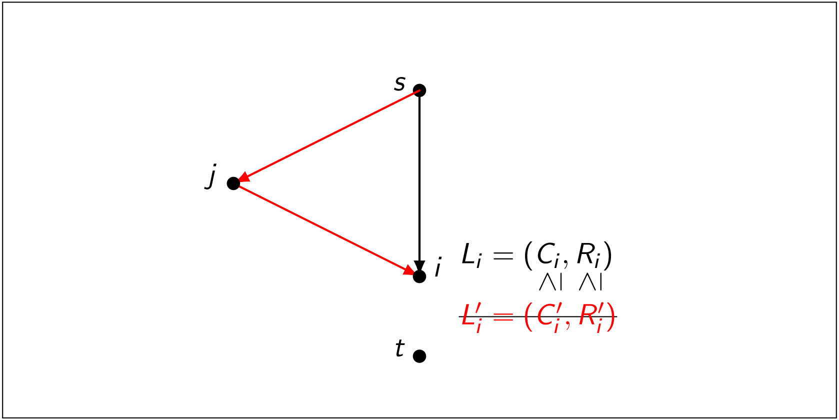
Exact Dynamic Programming for the ESPPRC

- We perform label extensions (C is the cost, R resource(s)):



Exact Dynamic Programming for the ESPPRC

- We eliminate dominated labels:



Exact Dynamic Programming for the ESPPRC

- We end when we performed all possible extensions:

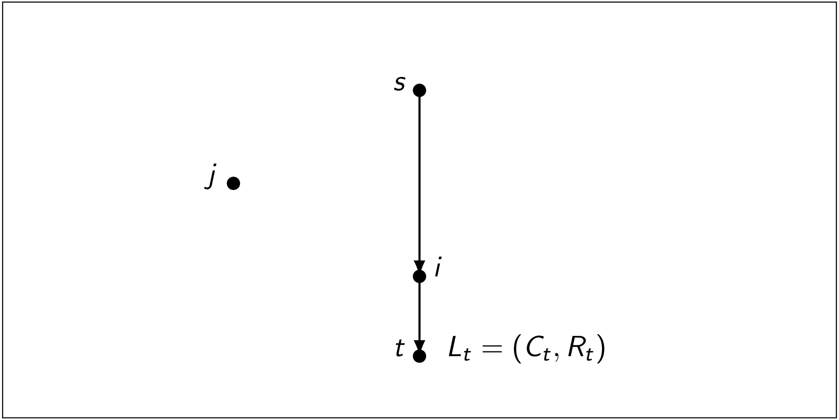


Table of Contents

ESPPRC model description

- For each vertex we have:
  - a time window $[a_i, b_i]$,
  - service time s_i,
  - delivery demand d_i,
  - a revenue (dual price) η_i.

ESPPRC model description

- For each vertex we have:
  - a time window $[a_i, b_i]$,
  - service time s_i,
  - delivery demand d_i,
  - a revenue (dual price) η_i.
- There is a single vehicle available at any time for a duration $\mathcal{S}$.

ESPPRC model description

- For each vertex we have:
  - a time window $[a_i, b_i]$,
  - service time s_i,
  - delivery demand d_i,
  - a revenue (dual price) η_i.
- There is a single vehicle available at any time for a duration $\mathcal{S}$.
- The total cost of a path P depends on total travel time T_P,the total of the collected dual prizes and the service start time T_s:

$$C_P(T_s) = T_P(T_s) - \sum_{i \in P} \eta_i.$$

Label structure for DP

- If we created a label in a straightforward manner we would use:
  - a service start time resource T_i, so that the state at i is feasible iff $T_i \in [a_i, b_i]$;

Label structure for DP

- If we created a label in a straightforward manner we would use:
  - a service start time resource T_i, so that the state at i is feasible iff $T_i \in [a_i, b_i]$;
  - a delivery demand resource Del_i, requiring $\text{Del}_i \in [0, Q]$;

Label structure for DP

- If we created a label in a straightforward manner we would use:
  - a service start time resource T_i, so that the state at i is feasible iff $T_i \in [a_i, b_i]$;
  - a delivery demand resource Del_i, requiring $\mathrm{Del}_i \in [0, Q]$;
  - a total spent time resource S_i, requiring $S_i \in [0, S]$;

Label structure for DP

- If we created a label in a straightforward manner we would use:
  - a service start time resource T_i, so that the state at i is feasible iff $T_i \in [a_i, b_i]$;
  - a delivery demand resource Del_i, requiring $\mathrm{Del}_i \in [0, Q]$;
  - a total spent time resource S_i, requiring $S_i \in [0, S]$;
  - an elementarity resource $(\mathrm{El}_k)^i_{k \in V}$, requiring $(\mathrm{El}_k)^i \in [0, 1], \ \forall k \in V$.

Label structure for DP

- If we created a label in a straightforward manner we would use:
  - a service start time resource T_i, so that the state at i is feasible iff $T_i \in [a_i, b_i]$;
  - a delivery demand resource Del_i, requiring $\mathrm{Del}_i \in [0, Q]$;
  - a total spent time resource S_i, requiring $S_i \in [0, S]$;
  - an elementarity resource $(\mathrm{El}_k)^i_{k \in V}$, requiring $(\mathrm{El}_k)^i \in [0, 1], \ \forall k \in V$.
- A DP state for vertex i in our scenario is therefore

$$(C_i, T_i, S_i, \mathrm{Del}_i, (\mathrm{El}_k)^i_{k \in V}).$$

Dominance rules: issue with time dependency

- T_i, S_i, and the total cost of the subpath s-i C_i clearly depend on the starting time T_s.

Dominance rules: issue with time dependency

- T_i, S_i, and the total cost of the subpath s-i C_i clearly depend on the starting time T_s.
- The DP state for i in our scenario then becomes

$$(C_i(T_s), T_i(T_s), S_i(T_s), \mathrm{Del}_i, (\mathrm{El}_k)^i_{k \in V}).$$

Dominance rules: issue with time dependency

- T_i, S_i, and the total cost of the subpath s-i C_i clearly depend on the starting time T_s.
- The DP state for i in our scenario then becomes

$$(C_i(T_s), T_i(T_s), S_i(T_s), \mathrm{Del}_i, (\mathrm{El}_k)^i_{k \in V}).$$

- We must therefore take into account an infinite number of Pareto-optimal states.

Dominance rules: issue with time dependency

- T_i, S_i, and the total cost of the subpath s-i C_i clearly depend on the starting time T_s.
- The DP state for i in our scenario then becomes

$$(C_i(T_s), T_i(T_s), S_i(T_s), \mathrm{Del}_i, (\mathrm{El}_k)^i_{k \in V}).$$

- We must therefore take into account an infinite number of Pareto-optimal states.
- To overcome this, we need only redefine the labels and the associated rules.

Time functions

- We can treat the time-dependent quantities by observing that the service start time at each node can be expressed with the recursion

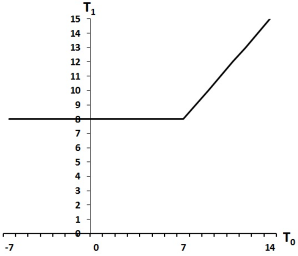
$$T_i(T_s) = \max\{a_i, T_{i-1}(T_s) + t_{i-1,i} + s_i\}$$

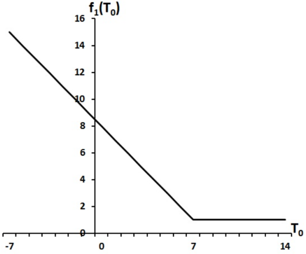
Time functions

- We can treat the time-dependent quantities by observing that the service start time at each node can be expressed with the recursion

$$T_i(T_s) = \max\{a_i, T_{i-1}(T_s) + t_{i-1,i} + s_i\}$$

- These are piecewise linear functions:





- Here $f_i(T_s)$ is the associated time-dependent cost.

New Dominance Rules and Resource Extension

- The new labels assume therefore the following structure:

$$L_i = \begin{cases} -l_i & = -\min\{l_{i-1}, b_i - \theta_i\} \\ \tilde{a}_i & = \max\{a_i, \tilde{a}_{i-1} + t_{i-1,i} + s_i\} \\ A_i & = \max\{A_{i-1} + t_{i-1,i} + s_i, \tilde{a}_i - l_i\} \\ \delta_i & = \delta_{i-1} - \eta_{i-1} \\ \mathrm{Del}_i & = \mathrm{Del}_{i-1} + d_i \\ \mathrm{El}_k^i & = \begin{cases} \mathrm{El}_k^{i-1} + 1 & \text{if} k = i \\ \mathrm{El}_k^{i-1} & \text{otherwise} \end{cases} \quad \forall k \in V \end{cases}$$

- Since we can properly define the extension and domination rules, we can apply all the existing improvements for the standard DP.

Bidirectional Dynamic Programming [6]

- To accelerate the procedure, we start it simultaneously from the sink, extending states *backwards*.

[5] Savelsbergh 1992.

[6] Introduced in Righini and Salani 2006.

Bidirectional Dynamic Programming [6]

- To accelerate the procedure, we start it simultaneously from the sink, extending states *backwards*.
- It suffices to invert the time windows with a constant M and change direction of the arcs, then use monodirectional DP:

$$[a_i, b_i] \Rightarrow [M - b_i, M - a_i], \quad (i, j) \Rightarrow (j, i)$$

[5] Savelsbergh 1992.

[6] Introduced in Righini and Salani 2006.

Bidirectional Dynamic Programming [6]

- To accelerate the procedure, we start it simultaneously from the sink, extending states *backwards*.
- It suffices to invert the time windows with a constant M and change direction of the arcs, then use monodirectional DP:

$$[a_i, b_i] \Rightarrow [M - b_i, M - a_i], \quad (i, j) \Rightarrow (j, i)$$

- The procedure is *bounded*: states are extended until the total amount of time spent is smaller than $S/2$, i.e. we consider total travel time as a *critical resource*.

[5] Savelsbergh 1992.

[6] Introduced in Righini and Salani 2006.

Bidirectional Dynamic Programming [6]

- To accelerate the procedure, we start it simultaneously from the sink, extending states *backwards*.
- It suffices to invert the time windows with a constant M and change direction of the arcs, then use monodirectional DP:

$$[a_i, b_i] \Rightarrow [M - b_i, M - a_i], \quad (i, j) \Rightarrow (j, i)$$

- The procedure is *bounded*: states are extended until the total amount of time spent is smaller than $S/2$, i.e. we consider total travel time as a *critical resource*.
- We need only to apply a *concatenation theorem*[5] to compute the actual total travel time T_P for each path P resulting from label concatenation.

[5] Savelsbergh 1992.

[6] Introduced in Righini and Salani 2006.

Table of Contents

DP improvements[7]: Duplicate Elimination

- During the phase of concatenation of forward and backward labels, the same path can be generated multiple times.

[7] Described in Righini and Salani 2008.

DP improvements[7]: Duplicate Elimination

- During the phase of concatenation of forward and backward labels, the same path can be generated multiple times.
- The path $P = s \rightarrow \cdots \rightarrow j \rightarrow i \rightarrow k \rightarrow \cdots \rightarrow t$ can be obtained by concatenating different pairs of labels, e.g. (l_i^{fw}, l_i^{bw}) or (l_j^{fw}, l_j^{bw}).

[7] Described in Righini and Salani 2008.

DP improvements[7]: Duplicate Elimination

- During the phase of concatenation of forward and backward labels, the same path can be generated multiple times.
- The path $P = s \to \cdots \to j \to i \to k \to \cdots \to t$ can be obtained by concatenating different pairs of labels, e.g. (l_i^{fw}, l_i^{bw}) or (l_j^{fw}, l_j^{bw}).
- Before each concatenation at i we check the forward and backward consumption of the critical resource, $R_{r,i}^{\mathrm{fw}}$ and $R_{r,i}^{\mathrm{bw}}$.

[7] Described in Righini and Salani 2008.

DP improvements[7]: Duplicate Elimination

- During the phase of concatenation of forward and backward labels, the same path can be generated multiple times.
- The path $P = s \rightarrow \cdots \rightarrow j \rightarrow i \rightarrow k \rightarrow \cdots \rightarrow t$ can be obtained by concatenating different pairs of labels, e.g. (l_i^{fw}, l_i^{bw}) or (l_j^{fw}, l_j^{bw}).
- Before each concatenation at i we check the forward and backward consumption of the critical resource, $R_{r,i}^{\text{fw}}$ and $R_{r,i}^{\text{bw}}$.
- We accept it only if they are as close as possible to half of the overall consumption of the resource along the path, i.e. iff $\Phi_i := |R_{r,i}^{\text{fw}} - R_{r,i}^{\text{bw}}|$ is minimum.

[7] Described in Righini and Salani 2008.

DP improvements[7]: Duplicate Elimination

- During the phase of concatenation of forward and backward labels, the same path can be generated multiple times.
- The path $P = s \to \cdots \to j \to i \to k \to \cdots \to t$ can be obtained by concatenating different pairs of labels, e.g. (l_i^{fw}, l_i^{bw}) or (l_j^{fw}, l_j^{bw}).
- Before each concatenation at i we check the forward and backward consumption of the critical resource, $R_{r,i}^{\text{fw}}$ and $R_{r,i}^{\text{bw}}$.
- We accept it only if they are as close as possible to half of the overall consumption of the resource along the path, i.e. iff $\Phi_i := |R_{r,i}^{\text{fw}} - R_{r,i}^{\text{bw}}|$ is minimum.
- The test is performed in constant time since we need only to check Φ_k if $R_{r,i}^{\text{fw}} < R_{r,i}^{\text{bw}}$ or Φ_j otherwise.

[7]Described in Righini and Salani 2008.

DP improvements: Decremental State Space Relaxation

- In **State Space Relaxation**[8] we project the state-space $\mathcal{S}$ used in DP to a lower dimensional space $\mathcal{T}$, so that the new states retain the cost.

[8] Developed by Christofides, Mingozzi, and Toth 1981.

[9] Developed by Righini and Salani 2009.

DP improvements: Decremental State Space Relaxation

- In **State Space Relaxation**[8] we project the state-space $\mathcal{S}$ used in DP to a lower dimensional space $\mathcal{T}$, so that the new states retain the cost.
- When applying this to the elementarity constraints, the number of states to explore is reduced, at the cost of feasibility.

[8] Developed by Christofides, Mingozzi, and Toth 1981.

[9] Developed by Righini and Salani 2009.

DP improvements: Decremental State Space Relaxation

- In **State Space Relaxation**[8] we project the state-space $\mathcal{S}$ used in DP to a lower dimensional space $\mathcal{T}$, so that the new states retain the cost.
- When applying this to the elementarity constraints, the number of states to explore is reduced, at the cost of feasibility.
- **Decremental**[9] State Space Relaxation (DSSR) is a generalization of both this method and DP with elementarity constraints.

[8] Developed by Christofides, Mingozzi, and Toth 1981.

[9] Developed by Righini and Salani 2009.

DP improvements: Decremental State Space Relaxation

- In **State Space Relaxation**[8] we project the state-space $\mathcal{S}$ used in DP to a lower dimensional space $\mathcal{T}$, so that the new states retain the cost.
- When applying this to the elementarity constraints, the number of states to explore is reduced, at the cost of feasibility.
- **Decremental**[9] State Space Relaxation (DSSR) is a generalization of both this method and DP with elementarity constraints.
- We maintain a set Θ of **critical** nodes on which the elementarity constraints are enforced at each iteration of DP.

[8] Developed by Christofides, Mingozzi, and Toth 1981.

[9] Developed by Righini and Salani 2009.

DP improvements: Decremental State Space Relaxation

- In **State Space Relaxation**[8] we project the state-space $\mathcal{S}$ used in DP to a lower dimensional space $\mathcal{T}$, so that the new states retain the cost.
- When applying this to the elementarity constraints, the number of states to explore is reduced, at the cost of feasibility.
- **Decremental**[9] State Space Relaxation (DSSR) is a generalization of both this method and DP with elementarity constraints.
- We maintain a set Θ of **critical** nodes on which the elementarity constraints are enforced at each iteration of DP.
- If at the end of DP the optimal path is not feasible, we update Θ with the nodes that are visited multiple times.

[8] Developed by Christofides, Mingozzi, and Toth 1981.

[9] Developed by Righini and Salani 2009.

DSSR strategies

- In the implementation of DSSR we can make decisions with regards to:

DSSR strategies

- In the implementation of DSSR we can make decisions with regards to:
    - initialization of the critical vertex set;

DSSR strategies

- In the implementation of DSSR we can make decisions with regards to:
  - initialization of the critical vertex set;
  - which vertices we insert in the set at the end of an iteration;

DSSR strategies

- In the implementation of DSSR we can make decisions with regards to:
    - initialization of the critical vertex set;
    - which vertices we insert in the set at the end of an iteration;
    - how many elementary paths we want to obtain for the CG procedure.

DSSR strategies

- In the implementation of DSSR we can make decisions with regards to:
  - initialization of the critical vertex set;
  - which vertices we insert in the set at the end of an iteration;
  - how many elementary paths we want to obtain for the CG procedure.
- These decisions involve trade-offs (e.g. cost of an iteration vs number of iterations).

DSSR strategies

- In the implementation of DSSR we can make decisions with regards to:
    - initialization of the critical vertex set;
    - which vertices we insert in the set at the end of an iteration;
    - how many elementary paths we want to obtain for the CG procedure.
- These decisions involve trade-offs (e.g. cost of an iteration vs number of iterations).
- We can associate parameters to these decisions, which we can then tune.

DP improvements: *ng*-route relaxation[10]

- Another elementarity relaxation approach aimed at increasing the number of dominated labels.

[10] Baldacci, Mingozzi, and Roberti 2011; Baldacci et al. 2010.

DP improvements: *ng*-route relaxation[10]

- Another elementarity relaxation approach aimed at increasing the number of dominated labels.
- For each i, we associate a neighbourhood of vertices N_i.

[10] Baldacci, Mingozzi, and Roberti 2011; Baldacci et al. 2010.

DP improvements: *ng*-route relaxation[10]

- Another elementarity relaxation approach aimed at increasing the number of dominated labels.
- For each i, we associate a neighbourhood of vertices N_i.
- We allow the creation of routes that include cycles of the form $i - \cdots - j - \cdots - i$ only if it contains a vertex j such that $i \notin N_j$.

[10] Baldacci, Mingozzi, and Roberti 2011; Baldacci et al. 2010.

DP improvements: *ng*-route relaxation[10]

- Another elementarity relaxation approach aimed at increasing the number of dominated labels.
- For each i, we associate a neighbourhood of vertices N_i.
- We allow the creation of routes that include cycles of the form $i - \cdots - j - \cdots - i$ only if it contains a vertex j such that $i \notin N_j$.
- We can explore several strategies to define the neighbourhoods, taking into account:
  - the distance between nodes,
  - the temporal distance of their time windows,
  - the difference in their demands,
  - whether we need neighbourhoods of fixed size or we prefer a cutoff distance,
  - whether we construct one neighbourhood for each node or if we rely on clusters.

[10] Baldacci, Mingozzi, and Roberti 2011; Baldacci et al. 2010.

Some preliminary results

Instance	normal BDP	DSSR	ng-r	ng-r + DSSR
1	0.887	0.695	0.637	0.736
2	43.168	5.857	13.693	5.788
3	182.677	21.8	29.196	14.112
4	424.655	47.8	75.602	38.73
5	12.61	1.728	3.506	1.597
6	10.396	7.914	3.605	7.152
7	90.007	43.723	24.89	37.649
8	358.448	128.768	70.814	95.971

Table of Contents

Matheuristics

- **Matheuristics** are 'heuristics algorithms made by the interoperation of metaheuristics and mathematical programming techniques'.[11]

[11] Boschetti et al. 2009.
[12] Archetti and Speranza 2014.

Matheuristics

- **Matheuristics** are 'heuristics algorithms made by the interoperation of metaheuristics and mathematical programming techniques'.[11]
- For routing problems, we can classify them in three classes[12].

[11] Boschetti et al. 2009.

[12] Archetti and Speranza 2014.

Matheuristics

- **Matheuristics** are 'heuristics algorithms made by the interoperation of metaheuristics and mathematical programming techniques'.[11]
- For routing problems, we can classify them in three classes[12].
  - **Decomposition approaches**: we identify subproblems that are solved independently, then combine their solutions. E.g. *Cluster first-route second* approaches.

[11] Boschetti et al. 2009.

[12] Archetti and Speranza 2014.

Matheuristics

- **Matheuristics** are 'heuristics algorithms made by the interoperation of metaheuristics and mathematical programming techniques'.[11]
- For routing problems, we can classify them in three classes[12].
  - **Decomposition approaches**: we identify subproblems that are solved independently, then combine their solutions. E.g. *Cluster first-route second* approaches.
  - **Improvement heuristics**: by solving a MILP, we improve an heuristic solution.

[11] Boschetti et al. 2009.

[12] Archetti and Speranza 2014.

Matheuristics

- **Matheuristics** are 'heuristics algorithms made by the interoperation of metaheuristics and mathematical programming techniques'.[11]
- For routing problems, we can classify them in three classes[12].
  - **Decomposition approaches**: we identify subproblems that are solved independently, then combine their solutions. E.g. *Cluster first-route second* approaches.
  - **Improvement heuristics**: by solving a MILP, we improve an heuristic solution.
  - **Branch-and-Price based approaches**, classified in *restricted master heuristics*, *heuristic branching* approaches, and *relaxation based* approaches.

[11] Boschetti et al. 2009.

[12] Archetti and Speranza 2014.

Restricted Master Heuristics

- The optimal solution of the master problem restricted to any subset of generated columns provides an heuristic solution.

[13] Joncour et al. 2010.

[14] Danna and Le Pape 2005.

Restricted Master Heuristics

- The optimal solution of the master problem restricted to any subset of generated columns provides an heuristic solution.
- The columns can either be generated heuristically or by solving exactly the pricing problem.

[13] Joncour et al. 2010.

[14] Danna and Le Pape 2005.

Restricted Master Heuristics

- The optimal solution of the master problem restricted to any subset of generated columns provides an heuristic solution.
- The columns can either be generated heuristically or by solving exactly the pricing problem.
- However, the master problem defined over a subset of columns is often infeasible[13], so we have to adopt techniques to recover feasibility or devise ways to obtain a suitable set of columns.

[13] Joncour et al. 2010.

[14] Danna and Le Pape 2005.

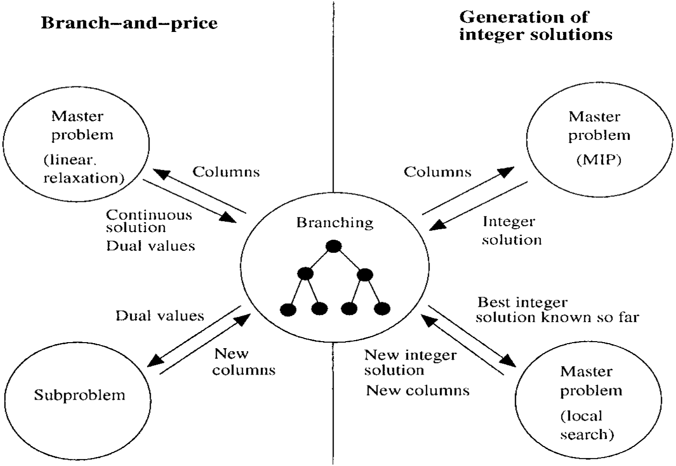
Restricted Master Heuristics

- The optimal solution of the master problem restricted to any subset of generated columns provides an heuristic solution.
- The columns can either be generated heuristically or by solving exactly the pricing problem.
- However, the master problem defined over a subset of columns is often infeasible[13], so we have to adopt techniques to recover feasibility or devise ways to obtain a suitable set of columns.
- Within the BP framework, we can use the RMH in a collaboration scheme with a metaheuristic[14], in order to obtain good solutions early in the procedure.

[13] Joncour et al. 2010.

[14] Danna and Le Pape 2005.

Collaboration scheme[15]



15 Image from Danna and Le Pape 2005.

Final remarks

- Possible future modification: *multi-trip* version.

[16] López-Ibáñez et al. 2011.

Final remarks

- Possible future modification: *multi-trip* version.
- Implementation of *automatic parameter tuning* with the aid of the `irace` package[16] developed by the Iridia team at the University of Bruxelles (Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle and Mauro Birattari).

[16] López-Ibáñez et al. 2011.

Final remarks

- Possible future modification: *multi-trip* version.
- Implementation of *automatic parameter tuning* with the aid of the `irace` package[16] developed by the Iridia team at the University of Bruxelles (Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle and Mauro Birattari).
- Feedback and suggestions are greatly appreciated!

[16] López-Ibáñez et al. 2011.

Thanks for your attention.

References I

- Claudia Archetti and M Grazia Speranza. "A survey on matheuristics for routing problems". In: *EURO Journal on Computational Optimization* 2.4 (2014), pp. 223–246.
- Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. "New route relaxation and pricing strategies for the vehicle routing problem". In: *Operations research* 59.5 (2011), pp. 1269–1283.
- Roberto Baldacci et al. "An exact solution framework for a broad class of vehicle routing problems". In: *Computational Management Science* 7.3 (2010), pp. 229–268.
- Andrea Bettinelli, Alberto Ceselli, and Giovanni Righini. "A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows". In: *Transportation Research Part C: Emerging Technologies* 19.5 (2011), pp. 723–740.
- Marco A Boschetti et al. "Matheuristics: Optimization, simulation and control". In: *Hybrid Metaheuristics*. Springer, 2009, pp. 171–177.
- Nicos Christofides, Aristide Mingozzi, and Paolo Toth. "State-space relaxation procedures for the computation of bounds to routing problems". In: *Networks* 11.2 (1981), pp. 145–164.

References II

- Emilie Danna and Claude Le Pape. "Branch-and-price heuristics: A case study on the vehicle routing problem with time windows". In: *Column Generation*. Springer, 2005, pp. 99–129.
- Guy Desaulniers and Daniel Villeneuve. "The shortest path problem with time windows and linear waiting costs". In: *Transportation Science* 34.3 (2000), pp. 312–319.
- Martin Desrochers and François Soumis. "A generalized permanent labeling algorithm for the shortest path problem with time windows". In: *INFOR Information Systems and Operational Research* (1988).
- Moshe Dror. "Note on the complexity of the shortest path models for column generation in VRPTW". In: *Operations Research* 42.5 (1994), pp. 977–978.
- Dominique Feillet et al. "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems". In: *Networks* 44.3 (2004), pp. 216–229.
- Cédric Joncour et al. "Column generation based primal heuristics". In: *Electronic Notes in Discrete Mathematics* 36 (2010), pp. 695–702.

References III

- Manuel López-Ibáñez et al. *The irace package, Iterated Race for Automatic Algorithm Configuration*. Tech. rep. TR/IRIDIA/2011-004. IRIDIA, Université Libre de Bruxelles, Belgium, 2011. URL: http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf.
- Giovanni Righini and Matteo Salani. "Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming". In: *Computers & operations research* 36.4 (2009), pp. 1191–1203.
- Giovanni Righini and Matteo Salani. "New dynamic programming algorithms for the resource constrained elementary shortest path problem". In: *Networks* 51.3 (2008), pp. 155–170.
- Giovanni Righini and Matteo Salani. "Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints". In: *Discrete Optimization* 3.3 (2006), pp. 255–273.
- Martin WP Savelsbergh. "The vehicle routing problem with time windows: Minimizing route duration". In: *ORSA journal on computing* 4.2 (1992), pp. 146–154.