Vehicle Routing: problems and methods

Tamás Kis
ELKH SZTAKI
kis.tamas@sztaki.hu

November 29, 2022

Overview

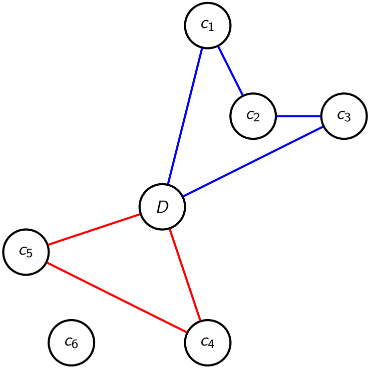
- What is Vehicle Routing?
- Offline problems:
  - VRP with time windows
  - Pickup-and-Delivery problems
- Online problems:
  - Stochastic Pickup-and-Delivery
  - The pickup-and-delivery challange of ICAPS2021

What is Vehicle Routing?

Input:

- Fleet of vehicles
- Depot(s)
- Clients
- Distance matrix
- Additional constraints
- Objective function

VRP: Find a set of tours that covers a subset of the clients, satisfies the constraints, and is optimal for the given objective function.



First concrete example: VRP with Time Windows

- V: set of clients and the depot.
- V_{cl}: set of clients, i.e., $V = V \setminus \{0\}$.
- for each $i \in V$:
  - $[e_i, d_i]$: time window of client i,
  - q_i: demand of client i.
- $q : V_{cl} \to \mathbb{R}_{>0}$: demand function of the clients.
- $G = (V, A)$ complete graph on V.
- Q: capacity of the vehicles.
- $c : A \to \mathbb{R}_{\geq 0}$: distance function on the edges, satisfies the triangle inequality.
- for $S \subset V$:

$$\delta^+(S) = \{(i,j) \in A \mid i \in S, j \in V \setminus S\},$$
$$\delta^-(S) = \{(i,j) \in A \mid i \in V \setminus S, j \in S\}.$$

VRP with time windows

- for $S \subset V_{cl}$: $r(S)$ is the minimum number of vehicles required to load the demand of the clients in S.
  - Computing $r(S)$ exactly requires the solution of a bin-packing problem.
  - Trivial lower bound: $\lceil \sum_{i \in S} q_i / Q \rceil$.
- Path $P = (v_1, \ldots, v_k)$ with arc set $A_P = \{(v_i, v_{i+1}) : i = 1 \ldots, k-1\}$. All vertices are different with the exception that both v_1 and v_k are the depot.
- Infeasible path: not possible to respect all its time windows while traversing the path.

MIP formulation of VRP with time windows

- For each $(i,j) \in A$, decision variable $x_{ij} \in \{0,1\}$: it is 1 if (i,j) is in a vehicle's route, and 0 otherwise.

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

$$\text{s.t.} \sum_{(i,k)\in\delta^-(k)} x_{ik} = 1, \quad k \in V_{cl} \tag{2}$$

$$\sum_{(k,j)\in\delta^+(k)} x_{kj} = 1, \quad k \in V_{cl} \tag{3}$$

$$x(\delta^-(S)) \geq r(S), \quad S \subseteq V_{cl} \tag{4}$$

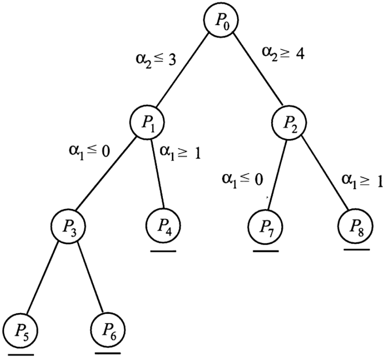
$$x(A_P) \leq |P| - 2, \quad \text{for each minimial infeasible path } P \tag{5}$$

$$x_{ij} \in \{0,1\}, \quad (i,j) \in A. \tag{6}$$

- The VRPTW polytope is the convex hull of points satisfying (2)-(6).

Linear programming based branch-and-bound

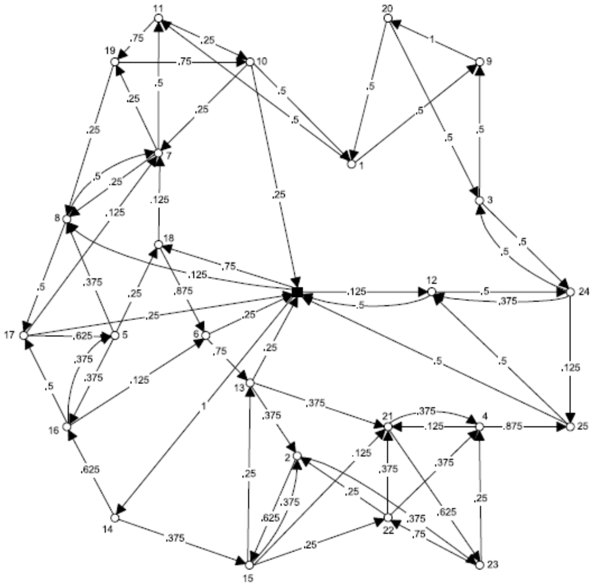
- LP relaxation: drop integrality of variables
- Solve LP relaxation, and if the solution is not-integral, branch on some variable.
- We get two subproblems, and repeat the above with them.



Solution by branch-and-cut

- Branch-and-bound + cutting planes
- Strengthen the LP relaxation by valid inequalities
- After solving the LP relaxation in a node, try to separate violated inequalities from different classes of valid inequalities

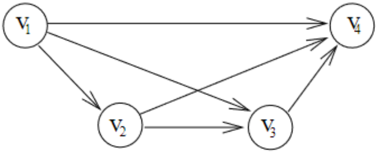
Example for LP-solution



Some classes of valid inequalities

For the infeasible path $P = (v_1, \ldots, v_k)$ the tournament constraint is

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} x_{v_i v_j} \leq k - 2. \qquad (7)$$



Proposition

The (7) is valid for the VRPTW polytope.

The reachability cuts of Lysgaard

- For any customer $i \in V_{cl}$, $A_i^- \subset A$ is the minimum arc set such that any feasible path $(0, \ldots, i)$ lies in A_i^-, i.e., $(j, k) \in A_i^-$ for each arc on the feasible path.
- To determine A_i^-, for each arc $(j, k) \in A$ check if there is a feasible path $(0, \ldots, j, k, \ldots, i)$.
- For any $S \subseteq V_{cl}$, and customer $i \in S$, define the $R_i^-(S)$ cut:

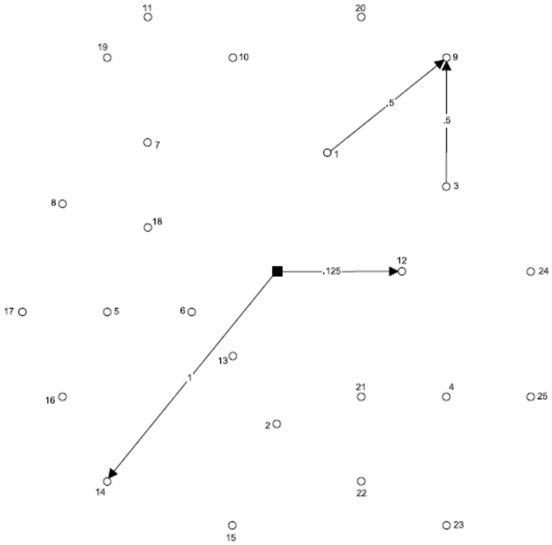
$$x(\delta^-(S) \cap A_i^-) \geq 1. \tag{8}$$

Proposition

The cut (8) is valid for the VRPTW polytope.

Example for $R_i^-(S)$ cut

For $S = \{1, 3, 9\}$, and $i = 9$, S is disconnected from the depot, and $x(\delta^-(S) \cap A_i^-) \geq 1$ is violated.



Reachability cuts

- For any customer $i \in V_{cl}$, $A_i^+ \subset A$ is the minimum arc set such that any feasible path $(i, \ldots, 0)$ lies in A_i^+, i.e., $(j, k) \in A_i^+$ for each arc on the feasible path.
- To determine A_i^+, for each arc $(j, k) \in A$ check if there is a feasible path $(i, \ldots, j, k, \ldots, 0)$.
- For any $S \subseteq V_{cl}$, and customer $i \in S$, define the $R_i^+(S)$ cut:

$$x(\delta^+(S) \cap A_i^+) \geq 1. \tag{9}$$

Proposition

The cut (9) is valid for the VRPTW polytope.

Reachability cuts for a subset of customers

- Customer set $T \subseteq V_{cl}$ is conflicting if and only if the customers in T must be served on $|T|$ separate routes in any feasible solution.
- For a conflicting customer set $T \subseteq V_{cl}$, the reaching arc set A_T^- is defined as $A_T^- := \bigcup_{i \in T} A_i^-$. The $R_T^-(S)$ cut is

$$x(\delta^-(S) \cap A_T^-) \geq |T|. \tag{10}$$

Proposition

The (10) cut is valid for the VRPTW polytope.

Reachability cuts for a subset of customers

- For a conflicting customer set $T \subseteq V_{cl}$, the reachable arc set A_T^+ is defined as $A_T^+ := \bigcup_{i \in T} A_i^+$. The $R_T^+(S)$ cut is

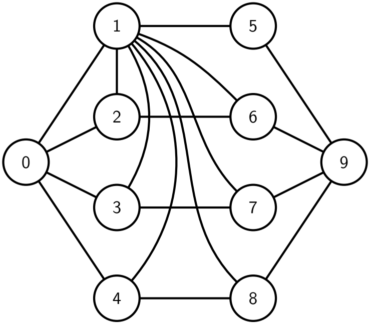
$$x(\delta^+(S) \cap A_T^+) \geq |T|. \tag{11}$$

Proposition

The cut (11) is valid for the VRPTW polytope.

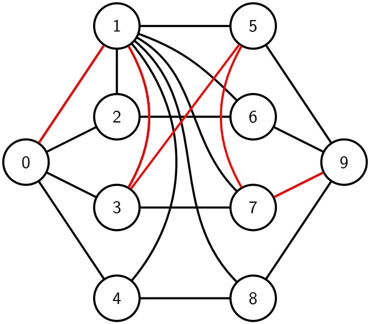
Pickup-and-Delivery Type Problems

- Instead of clients, there are pairs of pickup-and-delivery points
- From each pair, first the pickup point must be visited, then the delivery point.
- 0 and $2n+1$ are two copies of the depot.
- $P = \{1, \dots, n\}$ are the pickup points, and $D = \{n+1, \dots, 2n\}$ are the delivery points, and $(i, n+i)$ forms a pickup-delivery pair for $i \in P$.



Pickup-and-Delivery Type Problems

- Instead of clients, there are pairs of pickup-and-delivery points
- From each pair, first the pickup point must be visited, then the delivery point.
- 0 and $2n+1$ are two copies of the depot.
- $P = \{1, \ldots, n\}$ are the pickup points, and $D = \{n+1, \ldots, 2n\}$ are the delivery points, and $(i, n+i)$ forms a pickup-delivery pair for $i \in P$.



Modeling by a Mixed-Integer Program

$$\min \sum_{a \in A} c_a x_a \tag{12}$$

s.t.

$$\sum_{a \in \delta^+(0)} x_a = m \tag{13}$$

$$\sum_{a \in \delta^+(i)} x_a = 1 \qquad \forall i \in P \cup D \tag{14}$$

$$\sum_{a \in \delta^-(i)} x_a = 1 \qquad \forall i \in P \cup D \tag{15}$$

$$s_j - s_i \geq t_{ij} - M(1 - x_{ij}) \qquad \forall (i, j) \in A \tag{16}$$

$$s_{i+n} - s_i \leq T_{max} \qquad \forall i \in P \tag{17}$$

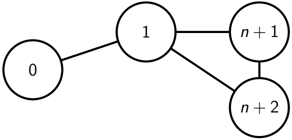
$$e_i \leq s_i \leq \ell_i \qquad \forall i \in P \cup D \tag{18}$$

Modeling by a Mixed-Integer Program (cnt.)

$$
\begin{aligned}
\sum_{(i,j)\in A: i,j\in S} x_{ij} &\leq |S| - 2 && \forall S \in \mathcal{S} && (19)\\
q_i + \sum_{a\in\delta^-(i)} b_a &= \sum_{a\in\delta^+(i)} b_a && \forall i \in P \cup D && (20)\\
0 \leq b_a &\leq Qx_a && \forall a \in A && (21)\\
b_{0i} = b_{i+n,2n+1} &= 0 && \forall i \in P && (22)\\
x_a &\in \{0,1\} && \forall a \in A && (23)
\end{aligned}
$$

where $\mathcal{S}$ consists of all the sets $S \subset N$ such that $0 \in S$, $2n+1 \notin S$, and $\exists\ i \in P$ such that $i \notin S$, but $i+n \in S$.

Example: $S = \{0, 1, n+1, n+2\}$.



Valid inequalities

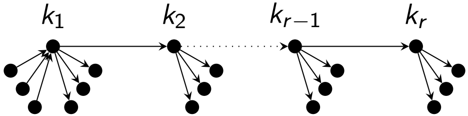
$\pi = (k_1, \ldots, k_r)$ compatible path,
$E_\pi = \{(k_i, k_{i+1}) \ : \ i = 1, \ldots, r-1\}$.

Proposition

For each $i \in \{1, \ldots, r\}$, let $S_1 \subseteq \delta^-(k_1)$ and $T_i \subseteq \delta^+(k_i) \setminus E_\pi$ such that the path $(s, k_1, ..., k_i, t)$ is incompatible for all $(s, k_1) \in S_1$ and $(k_i, t) \in T_i$ for any $1 \le i \le r$. The outfork constraint

$$\sum_{a \in S_1} x_a + \sum_{i=1}^{r} \sum_{a \in T_i} x_a + \sum_{i=1}^{r-1} x_{k_i, k_{i+1}} \le r \tag{24}$$

is valid for the DARP.



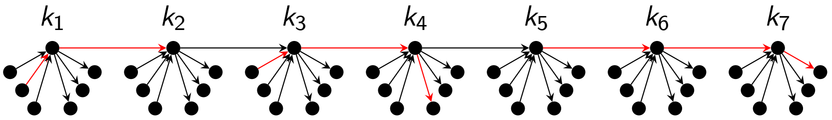
Fully lifted fork constraints

$\pi = (k_1, \ldots, k_r)$ compatible path,
$E_\pi = \{(k_i, k_{i+1}) \;:\; i = 1, \ldots, r-1\}$
$\forall i \in \{1, \ldots, r\}$: $S_i \subseteq \delta^-(k_i) \setminus E_\pi$ and $T_i \subseteq \delta^+(k_i) \setminus E_\pi$ such that the path $(s, k_i, ..., k_j, t)$ is not compatible for all $(s, k_i) \in S_i$ and $(k_j, t) \in T_j$ for any $1 \leq i \leq j \leq r$.

Proposition

The fully lifted fork constraint *is valid for the DARP.*

$$\sum_{i=1}^{r} \sum_{a \in S_i \cup T_i} x_a + \sum_{i=1}^{r-1} x_{i,i+1} \leq r. \tag{25}$$



Separation

$H = (S, T, E)$ bipartite graph, S and T color classes, where $S = \cup_{i=1}^{r} \delta^{-}(k_i) \setminus E_{\pi}$, $T = \cup_{i=1}^{r} \delta^{+}(k_i) \setminus E_{\pi}$.
For $a_i = (v_i, k_i) \in \delta^{-}(k_i)$ and $a_j = (k_j, v_j) \in \delta^{+}(k_j)$, the edge $\{a_i, a_j\}$ belongs to E for all $1 \leq i \leq j \leq r$ if and only if the path $(v_i, k_i, ..., k_j, v_j)$ is compatible.

Proposition

Let $U \subset S \cup T$ be a stable set in H. Then the system of sets $S_i = \delta^{-}(k_i) \cap U$ and $T_i = \delta^{+}(k_i) \cap U$ for all $i \in \{1, ..., r\}$ determines a fully lifted fork constraint.

Proof.

If the path $(v_i, k_i, ..., k_j, v_j)$ is compatible, then the two nodes corresponding to the arcs (v_i, k_i) and (k_j, v_j) are connected in the bipartite graph, so a stable set cannot contain both (v_i, k_i) and (k_j, v_j). $\square$

Separation (cnt.)

Algorithm

input: vector $x^\star$*, compatible path* $(k_1, \ldots, k_r)$*, and bipartite graph* H*.*
output: either a violated inequality or $\emptyset$*.*

1. *Weight each node* a *of* H *with* $x^\star_a$*.*
2. *Determine a maximum weight stable set* U *in* H*.*
3. *If* $x^\star(U) > r - \sum_{i=1}^{r-1} x^\star_{k_i,k_{i+1}}$*, then return* $x(U) + \sum_{i=1}^{r-1} x_{k_i,k_{i+1}} \le r$*, otherwise return* $\emptyset$*.*

Commodity cuts

For $i \in P$, let $S_i \subset \delta^-(i)$ and $S_{i+n} \subset \delta^-(i+n)$ be such that a_1 and a_2 are incompatible for all $a_1 \in S_i$ and $a_2 \in S_{i+n}$. The following inequality is valid for DARP:

$$\sum_{a \in S_i \cup S_{i+n}} x_a \leq 1. \tag{26}$$

Define analogously the inequalities

$$\sum_{a \in S_i \cup T_{i+n}} x_a \leq 1. \tag{27}$$

$$\sum_{a \in T_i \cup S_{i+n}} x_a \leq 1 \tag{28}$$

$$\sum_{a \in T_i \cup T_{i+n}} x_a \leq 1 \tag{29}$$

Computational results

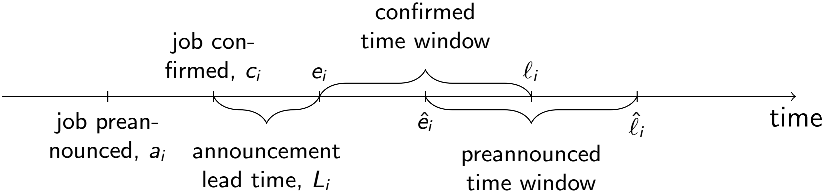
- CPLEX solver and implementation in C++
- CUT1 = some basic cuts and reachability cuts
- CUT2 = CUT1 + fully lifted fork constraints for $r \in \{1, 2\}$
- CUT3 = CUT2 + commodity cuts
- gap closed = $\frac{\text{"solution value"}}{\text{"best lower bound"}} - 1$

Comparisons

File	root gap			final gap			#nodes			#cuts		
	CUT_1	CUT_2	CUT_3	CUT_1	CUT_2	CUT_3	CUT_1	CUT_2	CUT_3	CUT_1	CUT_2	CUT_3
a2-16	0.0070	0	0	0	0	0	12	0	0	27	49	60
a2-20	0.0014	0	0	0	0	0	0	0	0	42	67	94
a2-24	0.0400	0.0031	0	0	0	0	1628	3	0	81	120	144
a3-18	0.0060	0	0	0	0	0	7	0	0	108	84	124
a3-24	0.0270	0	0	0	0	0	1486	0	0	347	185	162
a3-30	0.0003	0	0	0	0	0	3	0	0	269	341	240
a3-36	0.0198	0.0055	0	0	0	0	156	14	0	462	727	401
a4-16	0.0159	0.0040	0.0039	0	0	0	249	7	0	552	290	235
a4-24	0	0	0	0	0	0	0	0	0	122	169	161
a4-32	0.0273	0	0	0	0	0	7062	0	0	1750	689	586
a4-40	0.0492	0.0089	0.0045	0	0	0	140341	250	13	3366	1774	799
a4-48	0.0741	0.0143	0.0044	0.0484	0	0	137130	693	13	4733	4430	1105
a5-40	0.0484	0	0	0.0146	0	0	205485	0	0	2415	702	465
a5-50	0.0811	0.0105	0.0042	0.0607	0	0	92714	1090	70	7117	4357	1470
a5-60	0.0546	0.0104	0.0054	0.0352	0	0	49392	1183	224	5926	5412	2375
a6-48	0.1400	0.0088	0.0031	0.1140	0	0	82360	401	14	15643	6549	1847
a6-60	0.0407	0.0156	0.0022	0.0277	0	0	35503	667	9	16011	7558	2025
a6-72	0.0866	0.0168	0.0080	0.0706	0		29996	6476	675	9125	10637	4292
a7-56	0.0338	0.0113	0.0104	0.0227	0	0	42652	433	179	14577	5824	3062
a7-70	0.0553	0.0094	0.0043	0.0465	0	0	22315	2749	1232	19858	12642	6142
a7-84	0.0982	0.0220	0.0095	0.0899	0.0127	0	17955	8557	2061	19496	31022	7168
a8-64	0.0857	0.0189	0.0095	0.0768	0	0	29072	6768	427	12726	17933	3538
a8-80	0.1080	0.0217	0.0084	0.1000	0.0112	0	18182	5616	1378	17696	28579	6606
a8-96	0.1120	0.0336	0.0119	0.1070	0.0322	0.0018	8729	3100	8224	22672	38257	16760
$\sum$	1.213	0.215	0.0903	0.814	0.0569	0.0023	922429	38007	14519	175121	178397	59861

Dynamic vehicle routing problems

- Dynamic problems: transportation requests are not known in advance. Each request i has an preannouncement time a_i, a pickup location $p(i)$, a delivery location $d(i)$, and a preannounced time window $[\hat{e}_i, \hat{\ell}_i]$.
- After the preannouncement, transportation request i gets confirmed at some time $c_i > a_i$, and the confirmed time window is $[e_i, \ell_i]$.
- If request i is fulfilled, the profit earned is $profit_i$



Dynamic vehicle routing

- The vehicles serve one request at time, and after fulfilling a request, they proceed to the next request, or go back to the depot.
- The *routing cost* is the total distance traveled without serving a request.
- The objective is to minimize the routing cost $+$ the total profit of unserved (rejected) requests.

Formulation of the static problem

$$\text{minimize} \quad \sum_{(\alpha,\beta)\in E} cost_{\alpha,\beta} x_{\alpha,\beta} \tag{30}$$

subject to

$$x_{sv} = 1, \qquad \forall v \in V \tag{31}$$

$$\sum_{(\alpha,\beta)\in E} x_{\alpha,\beta} = \sum_{(\beta,\alpha)\in E} x_{\beta,\alpha}, \qquad \forall \alpha \in N \setminus \{s, t\} \tag{32}$$

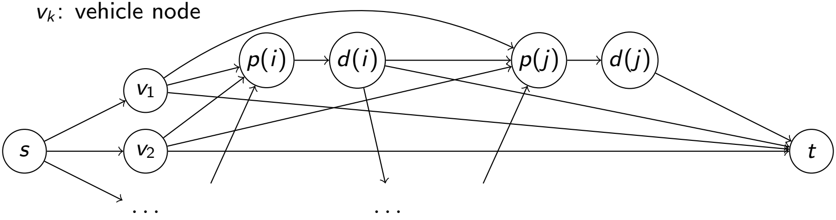
$$\max\{e_i, \tau_{depot,p(i)}\} \leq \delta_i \leq \ell_i, \qquad \forall i \in J \tag{33}$$

$$\delta_j + M(1 - x_{d(i),p(j)}) \geq \delta_i + \tau_{p(i),d(i)} + \tau_{d(i),p(j)}, \qquad \forall i, j \in J \tag{34}$$

$$x_{\alpha,\beta} \in \{0, 1\}. \qquad \forall (\alpha, \beta) \in E \tag{35}$$

Formulation of the static problem (cnt.)

$p(i)$, $d(i)$: pickup and drop-off nodes of customer i
v_k: vehicle node



$$cost_{\alpha,\beta} := \begin{cases} 0, & \text{if } (\alpha = s \text{ and } \beta \in V), \text{ or } (\alpha \in V \text{ and } \beta = t), \\ & \text{or, for some } i \in J,\ \alpha = p(i) \text{ and } \beta = d(i) \\ h \cdot dist_{depot,p(i)} - profit_i, & \text{if } \alpha \in V \text{ and } \beta = p(i) \text{ for some } i \in J \\ h \cdot dist_{d(j),p(i)} - profit_i, & \text{if } \alpha = d(j) \text{ and } \beta = p(i) \text{ for some } i \neq j \in J \\ h \cdot dist_{d(i),depot}, & \text{if } i \in J \text{ for some } i \in J,\ \alpha = d(i) \text{ and } \beta = t. \end{cases}$$

V = set of vehicle nodes, P = set of pickup nodes $p(i)$, D = set of delivery nodes $d(i)$

Stochastic information

- $\ell_i - e_i = \hat{\ell}_i - \hat{e}_i = TW_i$.
- There is a parameter L_i, the *lead time*, such that $e_i = c_i + L_i$.
- There exists $\Delta > 0$, the *range*, such that e_i is uniformly distributed in $[\hat{e}_i - \Delta, \hat{e}_i + \Delta]$.
- Consequently, $[e_i, \ell_i] \subset [\hat{e}_i - \Delta, \hat{\ell}_i + \Delta]$.

The Evenet Loop

Algorithm Event loop

Initialization: each vehicle is in the depot, no information is available about the customers.

1. Wait until a new event occurs (a customer preannounces/confirms its request or a vehicle arrives to its target location).
2. Invoke Subroutine Opt with the actual time t_{act}, the actual positions and states of the vehicles and the set J_{act} of not-yet-finished-or-rejected preannounced or confirmed requests received so-far.
3. According to the output of Subroutine Opt, send new commands to the vehicles.
4. If all customers are served or rejected, then the vehicles go back to the depot, and the processing of events is stopped. Otherwise, proceed with Step 1.

Subroutine Opt

Subroutine Opt

Input: actual time t_{act}, actual positions and states of the vehicles, confirmed information from each customer i with $c_i \leq t_{act}$, preannounced information from each customer with $a_i \leq t_{act}$.
Output: new actions for the vehicles

1. Build a minimum cost flow problem with respect to t_{act}.
2. Search an optimal (0/1) solution.
3. Determine $|V|$ (internally) node disjoint $s - t$ paths from the arcs with flow value 1 in the solution.
4. Determine the next action for each vehicle (according to the node that follows the vehicle node in a path).

Probabilistic model

- The arc costs on the arcs $(\alpha, p(j))$, where $\alpha \in V \cup D$ are redefined as

$$h \cdot dist_{\alpha,p(j)} - P(I_{\alpha,p(j)} = 1) \cdot profit_j.$$

- Random variable X_i represents the completion time of serving customer i.
- Random variable Y_j represents the end of the time window for serving customer j.
- If $\alpha = v$ for some $v \in V$, then

$$I_{v,p(j)} = \begin{cases} 1, & \text{if } t_{act} + \tau_{vj} \leq Y_j \\ 0, & \text{otherwise.} \end{cases}$$

$$P(I_{v,p(j)} = 1) := P(t_{act} + \tau_{vj} \leq Y_j).$$

Probabilistic model (cnt.)

If $\alpha = d(i)$ for some $i \in J_{act}$, then

$$I_{d(i),p(j)} = \begin{cases} 1, & \text{if } X_i + \tau_{ij} \leq Y_j \\ 0, & \text{otherwise.} \end{cases}$$

$$P(I_{d(i),p(j)} = 1) := P(X_i + \tau_{ij} \leq Y_j).$$

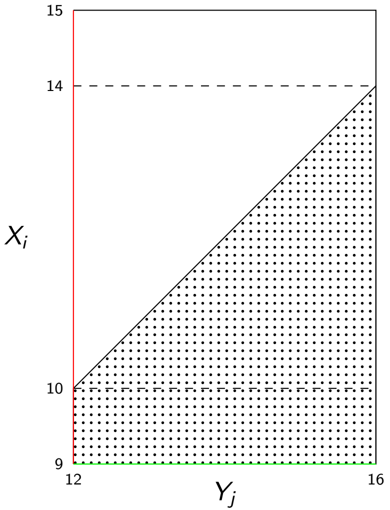
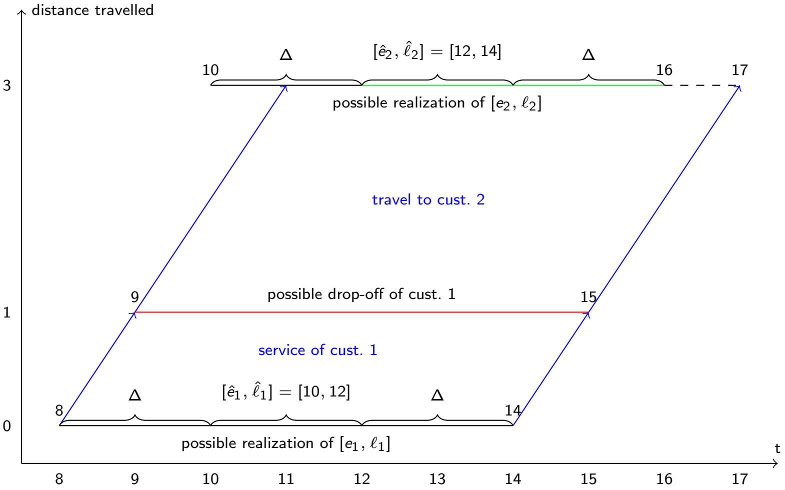


Illustration for computing the probabilities



Example

customer	a_i	$[\hat{e}_i, \hat{\ell}_i]$	pickup loc.	drop-off loc.	c_i	$[e_i, \ell_i]$
0	1	[22 23]	(3, 1)	(1, 2)	2	[27, 28]
1	1	[18, 19]	(0, 3)	(1, 1)	1	[18, 19]
2	2	[20, 21]	(2, 5)	(4, 1)	3	[20, 21]

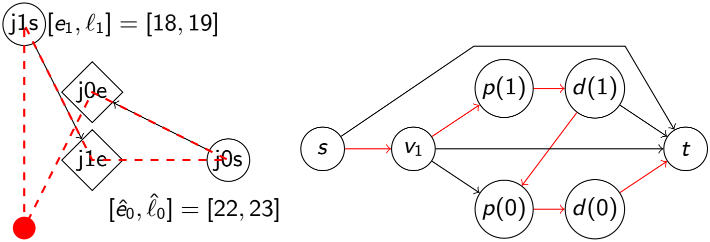
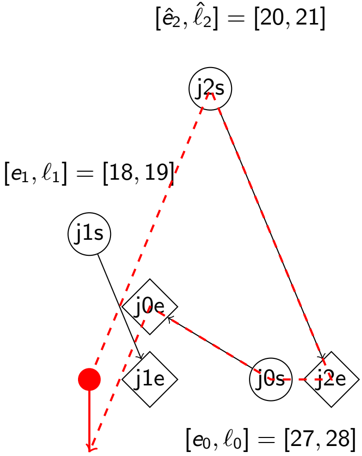
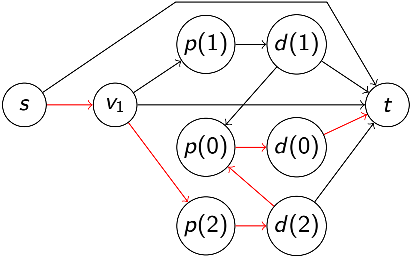


Figure: Left: known information at $t = 1$; the vehicle is at the red point (at $(0, 0)$). Right: the network flow problem at $t = 1$; the edges with flow value 1 in the optimal solution are red.

Example (cnt.)



$[\hat{e}_2, \hat{\ell}_2] = [20, 21]$
j2s
$[e_1, \ell_1] = [18, 19]$
j1s
j0e
j1e
j0s
j2e
$[e_0, \ell_0] = [27, 28]$



s
v_1
$p(1)$
$d(1)$
t
$p(0)$
$d(0)$
$p(2)$
$d(2)$

The MTS-seq method of Srour, Agatz and Oppen

- Maintain 60 scenarios and corresponding optimal routs of the vehicles.
- One scenario consists of randomly generated time windows for the not-yet-confirmed, but announced requests, and the confirmed time windows for the confirmed ones.
- A scenario is changed if
  - A time window is confirmed, or
  - t_{act} passes the beginning of a not-yet-confirmed, i.e., guessed, time window.

  When a scenario changes, a new optimal solution is computed by solving a MIP model.
- At any decision point, a route is chosen based on the 60 optimal solutions for 60 scenarios.

Computational results

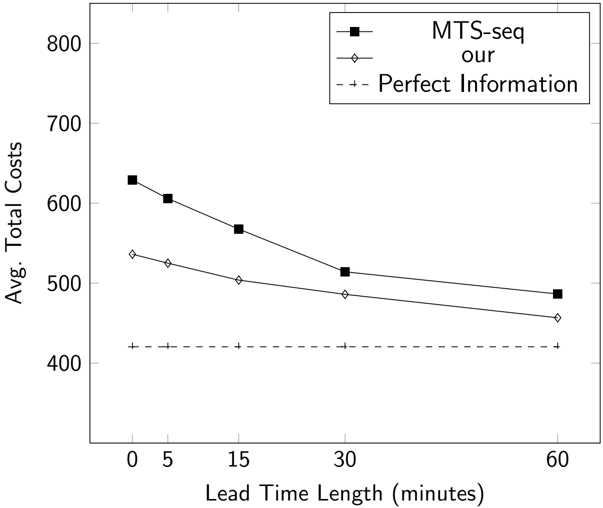


Figure: Dependence on Lead Time

Computational results (cnt.)

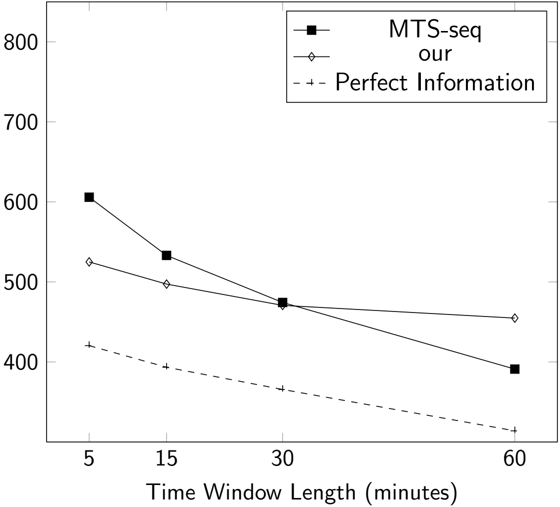


Figure: Dependence on Time Window Length

Computational results (cnt.)

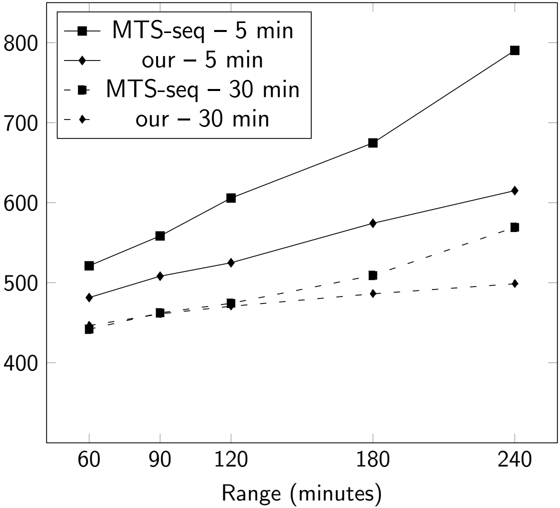
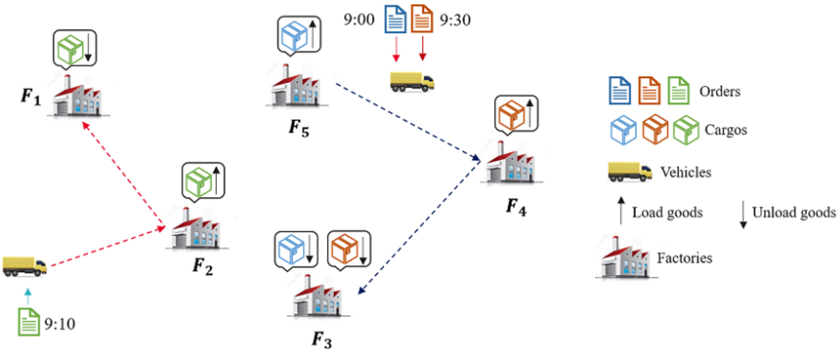


Figure: Dependence on Time Window Length and Range: $TWI_i \in \{5, 30\}$ minutes and $\Delta \in \{30, 45, 60, 90, 120\}$ mins.

The ICAPS 2021 competition

Dynamic Pickup and Delivery Problem specified by Huawei Ltd.

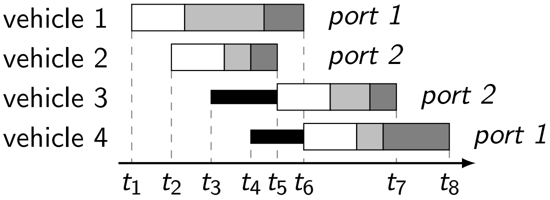


The Input

- Road network $G = (F, A, d)$, where F is the set of factories, A set of arcs, and $d = (d_{ij})$ distance matrix.
- Set of orders $O = \{o_i \mid i = 1, \ldots, N\}$, where $o_i = (F_p^i, F_d^i, q^i, t_e^i, t_l^i)$, where F_p^i and F_d^i are the pickup and delivery factories, $q^i = (q_{standar}^i, q_{small}^i, q_{box}^i)$ is the number of standard, and small pallets, and boxes of the shipment, and t_e^i, t_l^i is the creation time and due date of shipment, respectively.
- $V = \{v_k \mid k = 1, \ldots, K\}$ set of vehicles of the same capacity and speed.
- F set of factories, each factory has six cargo docks, where the vehicles can load and unload.
- Dock approaching time: 1800 seconds.
- Loading and unloading times: for q standard size pallets it is $180q$ seconds.

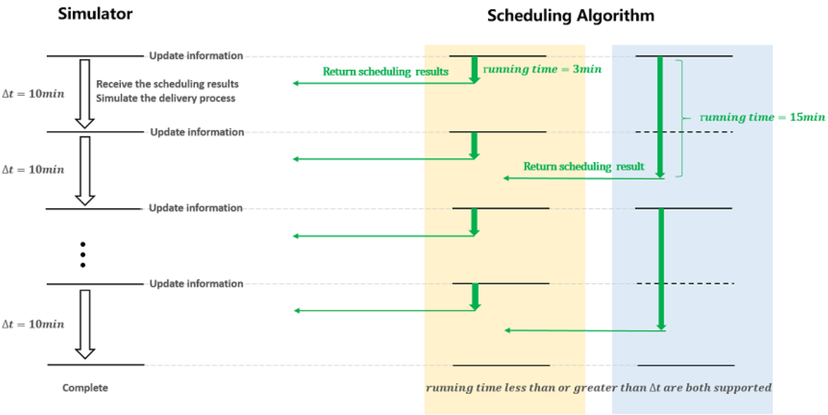
Constraints and objective

- All items must be fulfilled
- If an order o_i is not completed by t_l^i, then penalty is payed proportional to the delay.
- Orders that do not fit on a vehicle are split, the others are not.
- Last-In-First-Out sequence of load and unload. So, a route $(F_p^1, F_p^2, F_d^2, F_d^1)$ is feasible, but $(F_p^1, F_p^2, F_d^1, F_d^2)$ is not.
- The vehicles are served at the factories by the first-come-first-served rule.



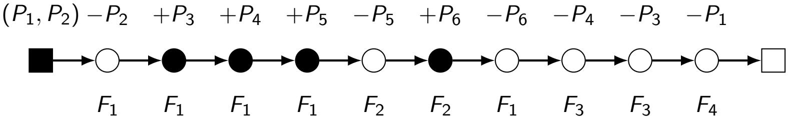
Simulation environment

- Provided by Huawei Ltd, implemented in Python.

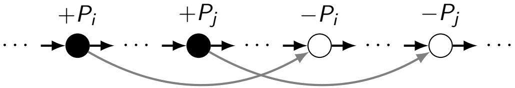


Solution of one epoch

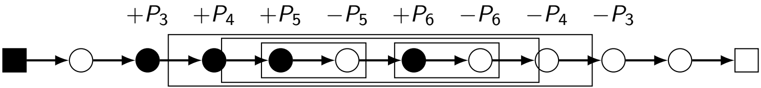
- Solution representation



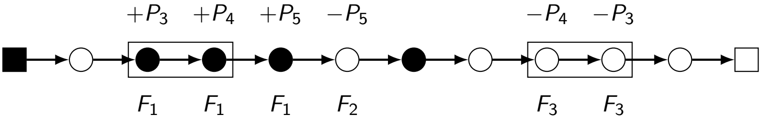
- Example for unfeasible route



- Blocks

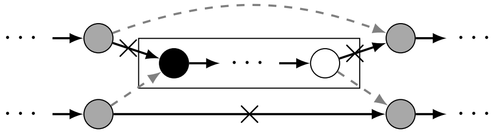


- Bridges

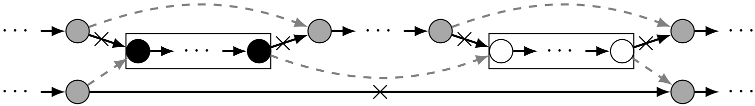


Neighborhood operations

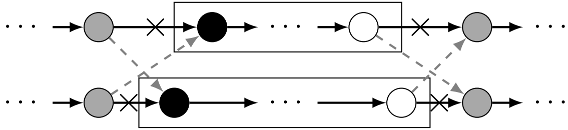
- Block relocation between routes



- Bridge relocation between routes. The two parts of the bridge will be adjacent after relocation.



- Block exchange between routes.



Problem instances

Table: Basic properties of public instances

Group	Instances	Orders	Vehicles
1	1 – 8	50	5
2	9 – 16	100	10
3	17 – 24	300	20
4	25 – 32	500	20
5	33 – 40	1000	50
6	41 – 48	2000	50
7	49 – 56	3000	100
8	57 – 64	4000	100

Comparison of different methods

Instances	LSBA*	1st Team[1]	2nd Team[2]	3rd Team[3]
Group 1	**1 306.7**	2 896.4	13 676.2	1 763.8
Group 2	**34 046.3**	41 535.3	-	62 180.2
Group 3	**687.8**	5 860.4	2 310.7	8 969.7
Group 4	6 831.4	**6 544.5**	105 049.4	26 938.3
Group 5	**10 344.8**	10 459.1	17 284.3	94 794.9
Group 6	42 249.8	**41 494.3**	153 419.1	651 944.9
Group 7	1 103 798.1	**798 240.7**	904 586.3	1 941 385.3
Group 8	**8 913 545.2**	11 359 466.4	18 678 529.1	15 122 816.2
All	**1 264 101.3**	1 533 312.1	-	2 238 849.2

* New local search-based approach
[1] Zhu et al. (2021)
[2] Ye and Liang (2021)
[3] Horváth el al. (2021)

Final remarks

- Vehicle routing problems are abundant
- Basic problems are still not solved satisfactorily
  - Need for faster exact methods
  - Need for better heuristics
- Dynamic problems are still not understood well
  - Need for a better understanding of the impact of client selection
  - Need for better understanding of route delay
  - In general: Need for a theory for dynamic VRP problems