Routing Problems with Loading Constraints
(with an introduction to Vehicle Routing)

Silvano Martello
DEI “Guglielmo Marconi”, Università di Bologna

Workshop on Traffic Optimization, Heidelberg, October 2015

This work by is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.
Summary

1. (Vertex) **Routing Problems**: A brief history and overview;

2. **Packing** (and Loading) **problems**:
   - One-, Two-, and Three-dimensional packing;

3. A combination: **Routing problems with loading constraints**.
1. Routing problems
The origins: circuits on graphs

SOLUTION

D'UN

QUESTION CURIEUSE QUI NE PAROIT
SOUMISE A AUCUNE ANALYSE,

PAR M. EULER.

Je me trouvais un jour dans une compagnie, où, à l'occasion du jeu
d'échecs quelqu'un proposa cette question: de parcourir avec un
cavalier toutes les cases d'un échiquier, sans revenir jamais deux fois à
la même, et en commençant par une case donnée. On mettoit pour cette
fin des jettons sur toutes les 64 cases de l'échiquier, à l'exception de celle où le Cavalier devait commencer sa route; et de chaque case
où le Cavalier passoit conformément à sa marche, on ôtoit le jetton,
de sorte qu'il s'agissoit d'enlever de cette façon successivement
 tous les jettons. Il falloit donc éviter d'un côté, que le cavalier ne re-
vint jamais à une case vide, et d'un autre côté il falloit diriger en
forte la course, qu'il parcourut enfin toutes les cases.

2. Ceux qui croyoient cette question affez aisée firent plusieurs
effas inutiles sans pouvoir atteindre au but; après quoi celui
qui avait propoé la question, ayant commencé par une case donnée,
a fui si bien diriger la route, qu'il a heureusement enlevé tous les jet-
tons. Cependant la multitude des cases ne permettoit pas qu'on ait
pu imprimer à la mémoire la route qu'il avait suivie; et ce n'étoit
qu'après plusieurs effais, que j'ai enfin rencontré une telle route, qui
satisfais à la question; encore ne valoit-elle que pour une certaine case
initiale. Je ne me souviens plus, si on lui a laissé la liberté de la choi-
sir lui-même; mais il a très positivement assuré qu'il étoit en état de
l'exécuter, quelle que soit la case où l'on voulut qu'il commençât.
• **Leonard Euler** (1759): *Solution d'une question curieuse qui ne paroit soumise à aucune analyse*.

  **Knights tour problem:**
  
  – a knight is placed on an empty $n \times n$ chessboard and must visit each square exactly once by only using valid chess moves of a knight:

  ![Chessboard](image1)

  ![Graph](image2)

  – by defining a graph in which the vertices correspond to the chessboard squares and the edges to the legal knight moves, a knight’s tour corresponds to a path (or a cycle) that visits every vertex of the graph exactly once:

  ![Graph](image3)
• Sir William Rowan Hamilton, Irish mathematician (1859): Icosian Game
  – played on a wooden planar representation of the edges of a dodecahedron (a graph), with holes at each of the twenty vertices:

  – The first player stuck five pegs in any consecutive vertices, and the second player was requested to stick the remaining fifteen pegs so as to complete the resulting path to a cycle visiting each vertex exactly once;

  – a circuit that passes through each vertex exactly once is called a Hamiltonian circuit;

  – sold for £25 to a Dublin toy manufacturer. (It seems that the sales were not satisfactory though.)
Hamiltonian circuits and the Traveling Salesman Problem

Given a directed graph $G = (V, A)$, $V = \{v_1, \ldots, v_n\}$, $A = \{(v_i, v_j) : v_i, v_j \in V\}$ or an undirected graph $G = (V, E)$, $E = \{(v_i, v_j) \equiv (v_j, v_i) : v_i, v_j \in V\}$ decide if the graph possesses a Hamiltonian circuit:

The problem can be solved by enumerating all permutations of the vertices and checking each of the for feasibility: This will take a time proportional to $(n - 1)!$ (exponential, impractical) but the problem is $\mathcal{NP}$-complete in the strong sense: most likely it will never be possible to solve it in a time that grows polynomially with $n$. 

---

S. Martello, Routing Problems with Loading Constraints
If the graph has weights (costs, lengths, times, ...) associated with the arcs/edges, the **Traveling Salesman Problem (TSP)** is to find the **Hamiltonian circuit of minimum total weight**:

![Graph with labeled edges and vertices]

cost = 16

Book published in 1931 in German: *The Traveling Salesman Problem, how he should be and what he should do to be successful in his business. By a veteran traveling salesman.*

Generalization of the Hamiltonian circuit problem $\implies \mathcal{NP}$-hard in the strong sense.

Applications in freight transportation using a **single vehicle**. **BUT**

Real world applications use a **fleet of vehicles** based at **one or more depots**, and have **specific constraints**.
Vehicle Routing Problems

- Optimal delivering of goods, using a set of vehicles, based at one or more depots, through a road network.

- **Freight transportation**
  - takes 10% - 25% of the final cost for consumer goods;
  - use of optimization techniques $\implies$ savings of 5% - 20% of the total transportation costs.

- General **Vehicle Routing Problem (VRP):** Find a set of routes, each assigned to a vehicle that starts and ends at its depot, so that a set of specific operational constraints is satisfied and the total transportation cost is minimized.

- Generalization of the Traveling salesman problem $\implies$ \( \mathcal{NP} \)-hard in the strong sense.

- Different models according to the considered constraints. Most models: **undirected graphs**

  Basic (simplest) VRP: **Capacitated Vehicle Routing Problem (CVRP):**
  - vertex 0=depot; vertices \( i (i = 1, \ldots, n) \)=customers; edge costs \( c_{ij} (i, j = 0, 1, \ldots, n) \);
  - customer \( i \) requests goods of total weight \( d_i (i = 1, \ldots, n) \);
  - each customer must be visited by a single vehicle;
  - \( K \) vehicles having capacity \( D \) (or having capacities \( D_k (k = 1, \ldots, K) \));
  - the total weight on each vehicle must be $\leq D$ (or on vehicle \( k \) must be $\leq D_k \ \forall \ k$);
  - minimize the total cost of the routes.
Vehicle Routing Problems (cont’d)

- **Distance-Constrained CVRP**: like CVRP, with an additional constraint:
  - $t_{ij} = \text{traveling time of edge } (i, j) \text{ (either coinciding with } c_{ij}, \text{ or not);} \]
  - $T_k = \text{maximum total traveling time for vehicle } k \text{ (} k = 1, \ldots, K\text{).} \]

- **VRP with Time Windows**: like CVRP, with an additional constraint:
  - $t_{ij} = \text{traveling time of edge } (i, j); \]
  - for each customer $i$,
    * $s_i = \text{time needed to download goods;} \]
    * $[a_i, b_i] = \text{time window within which the delivery must start.} \]

- **VRP with Backhauls**: like CVRP, with an additional constraint:
  - the customers are partitioned into two sets:
    *  
    * linehaul customers to whom goods are to be delivered; \]
    * backhaul customers whose goods need to be transported back to the depot; \]
  - each vehicle must visit all its linehaul customers before all its backhaul customers; \]
  - for each vehicle $k$,
    $\max \{(\text{total linehaul weight}), (\text{total backhaul weight})\} \leq D \text{ (or } \leq D_k \text{ } \forall \text{ } k). \]

Vehicle Routing Problems (cont’d)

- **VRP with Pickup and Delivery**: like CVRP, with an additional constraint:
  - for each customer $i$:
    * $d_i =$ quantity of goods to be delivered to the customer;
    * $p_i =$ quantity of goods to be picked up at the customer;
  - when the vehicle arrives, it first downloads then uploads;
  - for each vehicle $k$: the total carried weight at any time must be $\leq D$ (or $\leq D_k \quad \forall \; k$).

- **VRP with Loading Constraints**: like CVRP, with an additional constraint:
  - for each customer $i$:
    * list of the packages to be delivered, with their dimensions (height, width, length);
  - for each vehicle $k$: dimensions of the loading area (height, width, length);
  - the solution must provide, for each vehicle, the packing in the loading area.

- **Same constraints and variants studied for the TSP**
- **Combinations of the various constraints** (e.g., backhauls + time windows + ...)
- **Variants** (multiple depots, customers served by more vehicles, special compartments, ...)
- **All strongly $\mathcal{NP}$-hard.**
Bibliography

• **Hamiltonian circuits:**

• **Traveling salesman problem:**
  G Carpaneto, M Dell’Amico, P Toth. Exact solution of large-scale, asymmetric traveling salesman problems
  *ACM Transactions on Mathematical Software* (1995) freeware

• **Vehicle routing problems:**
2. Packing problems
One-dimensional bin packing problem

A geometrical description:

- given \( n \) segments (items) having width \( w_j (j = 1, \ldots, n) \), and an unlimited number of identical large segments (bins) having width \( W \),

\[
\begin{align*}
\text{pack all the segments,} & & \text{without overlapping, into the minimum number of bins:} \\
\end{align*}
\]

- The problem is \( \mathcal{NP} \)-hard in the strong sense.
- huge literature
Two-dimensional bin packing problem (2BP)

- given $n$ rectangles (items), having width $w_j$ and height $h_j$ ($j = 1, \ldots, n$),

- and an unlimited number of large rectangles (bins), having width $W$ and height $H$,

- A. pack all the items, without overlapping, into the minimum number of bins:

- B. pack a subset of items, without overlapping, in a single bin by maximizing the packed area:

- Many variants $\Leftarrow$ applications, cutting problems from standard stock pieces (wood, glass,...):
  - the items may/may not be rotated; $\Rightarrow$ by $90^\circ$/by any angle;
  - guillotine cutting may/may not be imposed (items must be obtained through a sequence of edge-to-edge cuts parallel to the edges of the bin);

- Generalizations of the One-Dimensional BP $\Rightarrow \mathcal{NP}$-hard in the strong sense.
Related multi-dimensional packing problems

- **Two-Dimensional Strip Packing Problem (2SP):**
  Pack a set of rectangular items into a strip of given width $W$ and infinite height, by minimizing the overall height of the packing. [Applications: cutting rolls of cloth/paper/...]

- **Three-Dimensional Bin Packing Problem (3BP):**
  Pack a set of rectangular boxes (items), of width $w_j$, height $h_j$ and length $l_j$ ($j = 1, \ldots, n$), into the minimum number of three-dimensional rectangular containers of width $W$, height $H$, and length $L$. [Applications: container loading, foam cutting]

- **Three-Dimensional Strip Packing Problem (3SP):**
  Pack a set of rectangular boxes into a strip of given width $W$, height $H$ and infinite length, by minimizing the overall length of the packing. [Applications: pallet loading]

- **2BP, 2SP, 3BP and 3SP** are strongly NP-hard, and very difficult to solve in practice.
Bibliography

• **One-dimensional bin packing**
  John Wiley & Sons, Chichester-New York (1990) freeware
  
  

• **Two-dimensional bin packing**

• **Three-dimensional bin packing**
  
  *ACM Transactions on Mathematical Software* 33 (2007) freeware
3. Routing problems with loading constraints
• Recall: **Capacitated Vehicle Routing Problem (CVRP):**
find at most \( K \) routes of minimum total cost to deliver goods demanded by a set of clients \( i \) (each requiring goods of total weight \( d_i \)), for a fleet of \( K \) vehicles of limited capacity \( D \), based at a central depot.

• **Applications to real world problems** limited by additional constraints:
  – **CVRP**: client demands = total weight of the items to be delivered;
  – **Real-world**: demands = sets of items with a weight and a shape \( \Rightarrow \) combination of CVRP with loading/packing problems.

• **2-Dimensional case:**
  – Transportation of rectangular-shaped items that **cannot** be stacked one on top of the other (big refrigerators, food trolleys, ...);
  – feasibility of packing on the truck bed;
  – feasibility of the loading and unloading operations.

• **3-Dimensional case:**
  – Transportation of rectangular-shaped boxes that **can** be stacked one on top of the other;
  – feasibility of box stacking (\( \Leftarrow \) fragility);
  – constraints on the stability of the loading;
  – feasibility of the loading and unloading operations.
CVRP + 2-Dimensional packing

- Complete undirected graph $G = (V, E)$: $V = \{0\}$ (depot) $\cup \{1, \ldots, n\}$ (clients); edge set $E = \{(i, j)\}$, with $c_{ij}$ = cost of edge $(i, j)$;

- $K$ identical vehicles, each having
  - weight capacity $D$;
  - rectangular loading surface of width $W$ and height $H$;

- demand of client $i$ ($i = 1, \ldots, n$):
  - $m_i$ items of total weight $d_i$;
  - item $I_{i\ell}$ ($\ell = 1, \ldots, m_i$) has width $w_{i\ell}$ and height $h_{i\ell}$;
  - the items must be orthogonally packed on the loading surface;

- each client must be served by a single vehicle;

- let $S(k) \subseteq \{1, \ldots, n\}$ be the set of clients served by vehicle $k$:
  - Weight constraint: total weight $\sum_{i \in S(k)} d_i \leq D$;
  - Loading constraint: there must be a feasible (non-overlapping) loading of all the transported items into the $W \times H$ loading area.
CVRP + 2-Dimensional packing (cont’d)

Objective:

• find a partition of the clients into at most \( K \) subsets and,

• \( \forall \) subset, a route starting and ending at the depot such that
  – all client demands are satisfied;
  – the weight constraint is satisfied;
  – the loading constraint is satisfied (feasible packing on the loading area);
  – the total cost of the edges is a minimum.

Two variants:

• **Unrestricted**: no further constraint;

• **Sequential**: the loading of each vehicle must be such that
  when a client is visited, the items of its lot can be downloaded through a sequence of straight movements (one per item) parallel to the \( H \)-edge of the loading area.
(a) Dashed strip = forbidden area for clients visited after client $i$
Sequential (b) and non-sequential (c,d) packings

Figura 1: Client 4 visited before Client 5
An instance with 3 vehicles and 8 clients (D = 100)
CVRP + 2-Dimensional packing: Tabu search

- **Neighborhood**
  - the algorithm can accept moves producing infeasible tours. Two infeasibilities:
    * **weight-infeasible**: total weight $> D$;
    * **load-infeasible**: height of the loading surface $> H$.
  - Infeasible moves are assigned a proportional **penalty**.

- **Feasibility check of the candidate tour:**
  - **weight-infeasibility**: immediate;

- **Tabu search objective function (infeasibilities = penalties):**
  - solution $s$ with $c(k) =$ total edge cost in route $k$:

  $$Z(s) = \sum_{k=1}^{K} c(k) + \alpha q(s) + \beta h(s)$$

  - $q(s) =$ total weight excess;
  - $h(s) =$ total height excess in the infeasible loadings;
  - $\alpha$ and $\beta =$ self-adjusting parameters.

- **Gendreau, Iori, Laporte, Martello, Networks (2010).**
CVRP + 2-Dimensional packing: Branch-and-Cut

ILP model:

- based on the classical two-index vehicle-flow formulation;
- only the sequential version has been addressed.

Infeasible routes:

- when an integer solution is found, every route is checked for weight and load feasibility;
- the checking procedure includes
  - greedy heuristics;
  - lower bounds;
  - branch-and-bound;
- ∀ infeasible route, a cut prohibiting it is added to the model;
- a pool of solved (proved feasible or infeasible) routes is stored.

CVRP + 2-Dimensional packing: Computational experiments

- CVRP instances from the literature with:
  - $15 \leq \# \text{ customers} \leq 255$;
  - $3 \leq \# \text{ vehicles} \leq 27$;
  - number and dimensions of items randomly generated according to five classes

- Branch-and-Cut:
solves to optimality instances with up to 25 customers

- Tabu Search:
  TS (5’ CPU limit) beats B&C (24 h CPU) for larger instances.

- Improve metaheuristic algorithms:
CVRP + 3-Dimensional packing

Same constraints as the 2-dimensional case, but

- the items are three-dimensional boxes;
- the boxes can be rotated by 90° degrees on the horizontal plane;
- some items can be fragile;
- no non-fragile item be placed over a fragile one;
- when boxes are stacked, the supporting surface must be large enough to guarantee stability;
- the loading of each vehicle must be such that when a client is visited, the items of its lot can be downloaded without shifting the items requested by other clients.
Figura 2: the vehicle is unloaded in the direction of the $z$ axis
An instance with 2 vehicles and 5 clients ($D = 100$)
CVRP + 3-Dimensional packing: Tabu search

- **Inner Tabu search**
  - iteratively invoked by the main Tabu search;
  - subproblem solved:
    - given an ordered set of clients to be visited in this order,
    - can all the requested items be feasibly loaded into a single vehicle?
  - the subproblem is NP-hard;
  - solved as a three-dimensional strip packing problem;
  - **neighborhood**: modify the loading sequence and execute two greedy heuristics;
  - if no feasible solution is found, the algorithm returns a 
    *load-infeasible* solution: loading length $\lambda > L$ (vehicle length);
  - if the total weight or the total volume exceeds the vehicle capacity, 
    *dummy loading*: length $\lambda = 2L$;
  - the main Tabu search uses $\lambda$ to evaluate the moves.

- **Outer Tabu search**: derived from the algorithm for CVRP + 2-Dimensional packing.
Other routing problems with loading constraints

- **Basis: Traveling Salesman Problem with Pickup and Delivery (TSPPD):**
  single vehicle must visit a set of customers, each associated with an origin location where some items must be picked up, and a destination location where such items must be delivered;
  find a shortest Hamiltonian cycle through all locations while ensuring that the pickup of any given request is performed before the corresponding delivery.
  
  - **TSPPD and LIFO loading:**
    pickups and deliveries must be performed in LIFO order (vehicles with a single access point);
  
  - **TSPPD and FIFO loading:**
    pickups and deliveries must be performed in FIFO order (AGVs that load items on one end and unload them at the other end);

- **CVRP + 2-dimensional loading + pickup and delivery constraints;**

- **3-dimensional container loading problems with multi-drop constraints** (special sequences);

- **CVRP with time windows and three-dimensional container loading;**

- **CVRP with pickup and delivery, delivery due dates and 3-dimensional loading:**
  auto-carrier transportation problem;

- **TSP with pickup and delivery and handling costs** (when the loading is not sequential);

- **...**
CVRP + 3-Dimensional packing: Computational experiments

Real-world instances:

- Italian company (furniture for bedrooms);
- fleet of private-owned vehicles paid per mileage;
- demands: three-dimensional rectangular items (to be assembled);
- identical vehicles (standard ISO containers);
- time windows neglected
- typical solutions: single day tours + multiple days tours.
- volumes between 1% and 4% of the vehicle volume;
- heights between 10% and 50% of the vehicle height.
A Tabu Search Algorithm for a Routing and Container Loading Problem

Italy are handled through a fleet of privately owned carriers. These carriers are paid by mileage, so the company is interested in minimizing the total distance traveled by its vehicles. These carriers are paid by mileage, so the company is interested in minimizing the total distance traveled by its vehicles. The value of the problem is the total distance traveled by the vehicles. This is an instance of the problem of minimizing the total distance traveled by the vehicles.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

In the multistart approach, the algorithm proved to be more sensitive to the parameters setting. A CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.

Concerning the single-start approach, a CPU time limit of 1,000 iterations was set. The number of iterations was set to be sensitive to the maximum number of iterations tolerated. The algorithm proved to be quite robust with respect to the starting values assigned to parameters.
<table>
<thead>
<tr>
<th>Instance</th>
<th>(n)</th>
<th>(M)</th>
<th>(K)</th>
<th>Greedy</th>
<th>1 hour CPU time</th>
<th>10 hours CPU time</th>
<th>24 hours CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(z)</td>
<td>(sec_z)</td>
<td>(z)</td>
</tr>
<tr>
<td>F01</td>
<td>44</td>
<td>141</td>
<td>4</td>
<td>7711</td>
<td>3723</td>
<td>2839.4</td>
<td>3694</td>
</tr>
<tr>
<td>F02</td>
<td>49</td>
<td>152</td>
<td>4</td>
<td>7167</td>
<td>4182</td>
<td>1993.8</td>
<td>4182</td>
</tr>
<tr>
<td>F03</td>
<td>55</td>
<td>171</td>
<td>4</td>
<td>6111</td>
<td>3674</td>
<td>3478.5</td>
<td>3650</td>
</tr>
<tr>
<td>F04</td>
<td>57</td>
<td>159</td>
<td>4</td>
<td>7059</td>
<td>4686</td>
<td>2520.5</td>
<td>4543</td>
</tr>
<tr>
<td>F05</td>
<td>64</td>
<td>181</td>
<td>4</td>
<td>7408</td>
<td>7235</td>
<td>2366.3</td>
<td>6886</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td>7091</td>
<td>4700</td>
<td>2639.7</td>
<td>4591</td>
</tr>
</tbody>
</table>

\(M = \) total number of items to deliver.
Additional Bibliography

• **Surveys and special issues:**


• **Recent results:**


**Conclusions and open perspectives**

Routing problems with loading constraints constitute a rich and lively research area.

Many issues need to be explored or better studied. For example:

- Evaluation of classical vehicle routing situations (e.g., allowing split deliveries).
- Heterogeneous vehicle fleets.
- Addition of special loading requirements (e.g., issues related to the center of gravity of the load).
- Use of column generation techniques for effectively determining exact solutions.
- Study of different objective functions reflecting special practical needs, such as routes with similar lengths or loads.
- Integration of these models and algorithms with location issues.
- ...
Thank You for your attention