Computers & Operations Research 38 (2011) 617–640

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

# A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle routing problem

Christophe Duhamel, Philippe Lacomme, Alain Quilliot, Hélène Toussaint *

*Laboratoire d'Informatique (LIMOS, UMR CNRS 6158), Campus des Cézeaux, 63177 Aubière Cedex, France*

ARTICLE INFO

*Keywords:*
Vehicle routing
GRASP
Iterated local search
Evolutionary local search
VRP
2L-CVRP

ABSTRACT

This paper addresses an extension of the capacitated vehicle routing problem where customer demand is composed of two-dimensional weighted items (2L-CVRP). The objective consists in designing a set of trips minimizing the total transportation cost with a homogenous fleet of vehicles based on a depot node. Items in each vehicle trip must satisfy the two-dimensional orthogonal packing constraints. A GRASP × ELS algorithm is proposed to compute solutions of a simpler problem in which the loading constraints are transformed into resource constrained project scheduling problem (RCPSP) constraints. We denote this relaxed problem RCPSP-CVRP. The optimization framework deals with RCPSP-CVRP and lastly RCPSP-CVRP solutions are transformed into 2L-CVRP solutions by solving a dedicated packing problem. The effectiveness of our approach is demonstrated through computational experiments including both classical CVRP and 2L-CVRP instances. Numerical experiments show that the GRASP × ELS approach outperforms all previously published methods.



## 1. Introduction

### 1.1. Capacitated vehicle routing problem

Keeping track of VRP development is strongly difficult because node routing problems subject matter transcends several academic disciplines. Lately Eksioglu et al. [1] in 2008 have provided a methodology to classify the literature of the VRP, i.e. a taxonomic framework. Their proposal extends the previous proposal of Current and Marsh [2] in 1993. Additional VRP constraints can be classified into three sets: scenario characteristics, problem physical characteristics and information characteristics. Scenario characteristics encompass, for instance, customer service demand quantity (deterministic, stochastic), load splitting constraints (splitting allowed or not), time windows (soft time windows, strict time windows), time horizon (single period, multi-period) or customer types (linehaul, backhaul, transfer). Problem physical characteristics encompass the number of origin points (single or multiple origins), the time window types (restrictions on customers, on roads), number of vehicles (exactly $n$ vehicles, limited number of vehicle, and unlimited number of vehicles). Information characteristics encompass the evolution of information (static or partially dynamic) or the quality of information (stochastic, deterministic).

The capacitated vehicle routing problem (CVRP) is a standard NP-hard node routing problem which received a considerable amount of attention for decades [3–5]. The CVRP consists in optimizing the delivery of goods required by a set of customers. It can be fully defined by considering a depot and a set of $n$ customers which correspond to the nodes of a complete graph $G=(V; E)$ where $V$ is a set of $n+1$ nodes, 0 being the depot and nodes $1,\ldots,n$ being the customers. Each edge $e \in E$ has a finite cost $c_e \geq 0$ and each node $v \in V-\{0\}$ has a demand $d_v \geq 0$. A fleet of homogenous vehicles of limited capacity $Q$ is based at the depot. The objective is to design a set of trips of minimal total cost to service all customers. A trip is a cycle performed by one vehicle, starting at the depot, ending at the depot and visiting a subset of nodes. The total load trip is upper bounded by the vehicle capacity $Q$. Since split deliveries are not allowed, each customer is serviced by one vehicle only. Medium and large scale CVRP instances resolution is limited to metaheuristics as stressed in [6].

### 1.2. Capacitated vehicle routing problem with packing: 2L-CVRP

The 2L-CVRP is an extension of the CVRP which includes two-dimensional rectangle loading constraints (the 2L constraints). This problem, first introduced in [7–9], can be reduced to the CVRP when dropping the size of the items or when considering $1 \times 1$ square

* Corresponding author.
*E-mail address:* helene.toussaint@isima.fr (H. Toussaint).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.08.017

**Nomenclature**

*Problem definition*

| | |
|---|---|
| $G=(V;E)$ | a complete undirected graph |
| $V$ | set of $n+1$ nodes with 0 the depot node |
| $c_{ij}$ | cost from node $i$ to $j$ |
| $n$ | number of nodes (a node is a customer) |
| $N$ | number of vehicles |
| $D$ | weight capacity of vehicles |
| $W$ | vehicle width |
| $L$ | vehicle length |
| $A=W\times L$ | vehicle area |
| $m_i$ | number of items to deliver at customer $i$ |
| $d_i$ | total weight of items to service at customer $i$ |
| $w_{ik}$ | item width of item $k$ at customer $i$ |
| $l_{ik}$ | item length of item $k$ at customer $i$ |

*Framework parameters*

| | |
|---|---|
| $p$ | percent of vehicle volume used during heuristics runs |
| $nb$ | number of solutions kept during the grasp process |
| $np$ | number of GRASP iterations (number of initial solutions investigated) |
| $ns$ | number of ILS iterations |
| $nd$ | number of parallel mutation/local search |
| $nr$ | maximum number of iterations without improvement per ELS |
| $ne$ | number of attempts to generated a initial feasible solution |
| $\varepsilon$ | a small real number representing the threshold required to continue the RL |

*Solution*

| | |
|---|---|
| $T$ | a giant tour |
| $T=(v_1, \ldots, v_n)$ | sequence customer in the giant tour $T$ |
| $t$ | a trip |
| $n(t)$ | number of customers in trip $t$ |
| $t=(t_0, t_1, \ldots, t_{n(t)}, t_{n(t)+1})$ | sequence customers in trip $t$ |
| $S$ | a RCPSP-CVRP solution (set of trips) |
| $f(S)$ | cost of $S$ |
| $N(S)$ | number of trips in solution $S$ |
| $\alpha$ | penalty |
| $t(S)$ | set of trips of solution $S$ |
| $f(t)$ | cost of the trip $t$ |
| $S^*$ | best RCPSP-CVRP solution found |
| $f^*$ | cost of $S^*$ |
| $O$ | cost ordered set of RCPSP-CVRP solutions |

*RCPSP*

| | |
|---|---|
| $na$ | number of activities to schedule |
| $m$ | number of resources |
| $i/j$ | activity to schedule |
| $d_i$ | duration of activity $i$ |
| $r_{ik}$ | requirement of activity $i$ for resource $k$ |
| $u$ | sink activity |
| $s$ | source activity |
| $ES_i$ | earliest starting time of activity $i$ |
| $LS_i$ | latest starting time of activity $i$ |

*Split*

| | |
|---|---|
| $H_T=(X, Y, Z)$ | auxiliary digraph linked to the giant trip $T$ |
| $X$ | set of $n+1$ nodes |
| $Y$ | set of arcs in $H$ where arc $(i, j)$ represent a trip servicing customers $v_{i+1}$ to $v_j$ |
| $Z_{ij}$ | trip cost link to the arc $(i, j)$ |

*Bin packing*

| | |
|---|---|
| $W\times L$ | bin size |
| $w_i\times l_i$ | item size |
| $ni$ | number of items |
| $D_y$ | set of items $i$ which can be scheduled at position $(x_i, y)$ |
| $x_i$ | x-position of item $i$ (either $ES_i$ or $LS_i$) |

*Hash function/map*

| | |
|---|---|
| $K$ | a huge number used by the hash function |
| $h(t)$ | hash value for trip $t$ |
| $F(h(t))$ | function giving in O(1) the RCPSP feasibility of trip $t$ with hash function $h(t)$ |

items, thus dealing only with their weight. This is strongly relevant to distribution companies since it combines both vehicle routing optimization and two-dimensional items packing.

More formally, each vehicle of the homogenous fleet is now defined by a weight capacity $D$ and by a rectangular two-dimensional loading area $A=W\times L$, where $W$ is the vehicle width and $L$ is the vehicle length. The demand of each customer $i=1,\ldots,n$ consists in a set of $m_i$ items of total weight $d_i$: each item $k=1,\ldots,m_i$ has width $w_{ik}$ and length $l_{ik}$. Each customer must be serviced by only one vehicle, which is assigned to a single trip. A trip $t$ is a sequence $t=(t_0, t_1,\ldots,t_{n(t)}, t_{n(t)+1})$ of customers where $t_0=t_{n(t)}+1$ corresponds to the depot. It must be simultaneously "weight-feasible" and "packing-feasible". A trip $t$ is stated "weight-feasible" if the total weight does not exceed the vehicle capacity, i.e. $\sum_{i\in t} d_i \leq D$ and it is stated "packing-feasible" if the customer items can be loaded without overlapping into the vehicle and satisfying the classical packing constraints. A set of "weight-feasible" and "packing-feasible" trips defines a solution of the 2L-CVRP.

According to Fuellerer et al. [9] classification, four different cases can be distinguished with respect to the loading configurations. To prevent ambiguities between the notations of Gendreau et al., Zachariadis et al., and lately by Fuellerer et al., we propose the notation $x|yz|L$ where $x$ represent the dimension (two dimensional or three dimensional), $y$ represents the items order constraint (sequential or unrestricted) and $z$ represents the items orientation (oriented or rotated). Four two-dimensional problems can be defined:

- 2|SO|L: two dimensional sequential oriented loading
- 2|UO|L: two dimensional unrestricted oriented loading
- 2|SR|L: two dimensional sequential rotated loading
- 2|UR|L: two dimensional unrestricted rotated loading

In a "Sequential" problem items must be packed into the vehicle in such a way that unloading the items for each customer in the trip can be achieved through a sequence of straight movements (one per item). This additional constraint ensures that no item required by a customer serviced afterwards prevents an item of the current customer to be unloaded. "Unrestricted" means that there is no restriction in the items packing Problem, i.e. one item unload could require several costly movements of items. In "Oriented" problems no rotation of items are possible while they are allowed in "Rotated" problems.

The 2L-CVRP resolution has been first addressed by Iori et al. [10] using an branch and cut approach limited to small scale instances (less than 25 customers) dedicated to sequential oriented loading. Then Gendreau et al. [7] introduced a tabu search algorithm for both sequential and unrestricted large scale instances. To the best of our knowledge, the Ant Colony scheme introduced by Fuellerer et al. [9] is the most efficient approach to solve the 2L-CVRP. Three dimensional loading CVRP (3L-CVRP) have been recently addressed by Gendreau et al. but only small scale instances are tested since three dimensional packing problems are much more difficult than two dimensional ones.

### 1.3. Cutting and packing problems

Packing problems belong to the well-known family of *cutting and packing problems*. Many packing problems deal with the insertion of rectangular items in rectangular bin. They mostly differ on the objective function to minimize.

- The two-dimensional bin packing problem (2BPP) consists in packing a set of rectangular items into a minimum number of identical rectangular bins.
- The two-dimensional strip packing problem (2SP) consists in packing a set of rectangular items into a strip of known width and infinite height so as to minimize the overall height of the packing.
- The two-dimensional orthogonal packing problem (2OPP) consists in determining if a set of rectangular items can be packed into one bin (rectangle) of fixed size.

Several extensions have been tackled over time in scientific publications, including but not limited to rotation of items, limitations on the total weight and/or item costs. The 2L-CVRP packing problem falls into the last category since the objective for a trip is to be sure that items can be packed into the vehicle.

A 2OPP instance consist in a set $I=\{1,\ldots,n\}$ of items which have to be packed and of a bin $B=(L;W)$ fully defined by its length $L$ and its width $W$. An item $i$ has a length $l_i$ and a width $w_i$ ($l_i$, $w_i \in IN$). A solution of the problem consists in defining the position of each item $i$ (denoted by $(x_i; y_i)$ and corresponding to the coordinates of its bottom left-hand corner) without overlapping.

Several exact methods are described in literature for the 2OPP including, methods which pack items one by one [11,12], methods promoting constraint programming techniques [13,14], methods taking advantages of graph theory [15,16] and methods addressing a relaxed problem. Exact resolution schemes are time consuming and then limited to small and medium scale instances with less than 20 items to pack. Large scale instances have been efficiently addressed by heuristic and metaheuristic schemes based on simulated annealing (see for example [17]) or genetic algorithms (see [18,19] for example). In fine, packing problems resolution is one of the challenging problems to solve when addressing 2L-CVRP: the difficulty mostly comes from a great part, of the huge number of constraints generated by the items geometry. A 2OPP example is introduced below including a graphical solution representation.

Let us consider a 2OPP instance with 6 items (Table 1) which must be packed into a bin $B=(10;5)$. Table 1 gives one 2OPP solution, i.e. the position $(x_i, y_i)$ of each item in the bin. Fig. 1 gives a graphical representation of the 2OPP solution described in Table 2.

**Table 1**
An instance of 2OPP.

| Item $i$ | Length $l_i$ | Width $w_i$ |
|---|---|---|
| A | 4 | 2 |
| B | 2 | 5 |
| C | 3 | 1 |
| D | 1 | 2 |
| E | 2 | 3 |
| F | 3 | 3 |

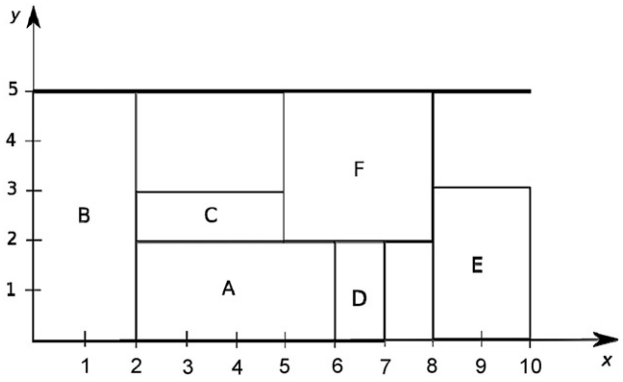


**Fig. 1.** One 2OPP graphical representation solution.

**Table 2**
One 2OPP solution.

| Item $i$ | $x_i$ | $y_i$ |
|---|---|---|
| A | 2 | 0 |
| B | 0 | 0 |
| C | 2 | 2 |
| D | 6 | 0 |
| E | 8 | 0 |
| F | 5 | 2 |

**Table 3**
An instance of RCPSP.

| Activity $j$ | Duration $d_j$ | Resource $r_j$ |
|---|---|---|
| A | 4 | 2 |
| B | 2 | 5 |
| C | 3 | 1 |
| D | 1 | 2 |
| E | 2 | 3 |
| F | 3 | 3 |

**Table 4**
One RCPSP solution.

| Activity $j$ | $x_j$ |
|---|---|
| A | 2 |
| B | 0 |
| C | 2 |
| D | 6 |
| E | 8 |
| F | 5 |

### 1.4. Resource-constrained project scheduling problem: RCPSP

The resource-constrained project scheduling problem (RCPSP) is composed of a set of *na* activities and a set of *m* resources. The terminology is quite different from the 2OPP terminology since "item" is replaced by activity to illustrate there is no geometric consideration in activity definition. Each activity $i$ is characterized by its duration $d_i$ and his requirement $r_{ik}$, $k=1,\ldots,m$, in resources. Activities are interrelated by precedence constraints which state that one activity $j$ cannot start before its immediate predecessors have been achieved. For the sake of simplicity, a unique source activity $s$ and a unique sink activity $u$ are usually included in the project. They correspond to the "project start" and to the "project end", respectively. The aim of the RCPSP is to schedule all activities satisfying both precedence and resource constraints and minimizing the total project duration (the makespan). The structure of the project is usually modeled by a so-called activity-on-node (AON) network where the nodes represent the activities and the arcs represent the precedence constraints.

The RCPSP is a challenging problem of great interest that has been widely studied over the past decades. Since it is an extension of the job-shop, it is NP-hard (see [20,21] for details on complexity). Several surveys are available including the survey of Herroelen et al. [22], of Kolisch and Padman [23], of Weglarz [24] and of Demeulemeester and Herroelen [25]. Kolisch and Padman [23] also surveyed some heuristic methods for classes of project scheduling problems. Heuristic-based approaches are completed by numerous iterative improvement schemes including Memetic Algorithm, Tabu Search for instance. Note that an efficient insertion technique has been proposed by Artigues et al. [26]. Tseng and Chen [27] provided detailed experiments on methods taking into account the computation time and other performance criteria. A RCPSP solution is fully defined by the starting time of each activity.

The RCPSP solution only guaranties that the total amount of resources satisfies the total consumption of activities, at any time.

For the instance of Table 3, we provide the RCPSP solution of Table 4 with the starting time $x_i$ of all activities $i$.

An example of RCPSP instance is provided in Table 3. The RCPSP instance is composed of six activities and five units of a single resource.

In RCPSP, the resource consumption over time depends of the starting time $x_i$ of activity. Fig. 2 gives the resource consumption linked to the solution of Table 4.

### 1.5. RCPSP and two orthogonal packing problem

Table 5 sums up the similarities between the two problems for both objective and solutions required. This table highlights that a 2OPP solution consists in defining for each item $i$ a pair $(x_i, y_i)$, while a RCPSP solution consists in defining only a starting time $x_i$ for each activity $i$. Hartmann [28] stressed that packing and project scheduling problems are completely different with respect to their

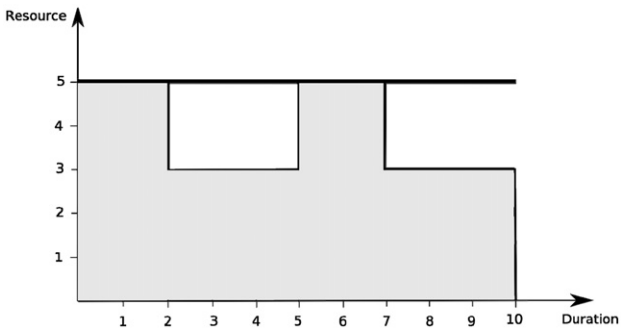


**Fig. 2.** One RCPSP graphical representation of the resource consumption.

**Table 5**
RCPSP vs. 2OPP.

| | **2OPP** | **RCPSP** |
|---|---|---|
| **Problem statement** | Pack items into one bin | Schedule activities |
| **Object** | 2 dimensional items defined by length and width | Activities defined by resources requirement and duration |
| | Items | Activities |
| | Length | Duration |
| | Width | Resource requirement |
| **Solution** | For each item i, position $(x_i, y_i)$ | for each activity $i$, starting time $x_i$ |

applications but it is possible to compare the mathematical properties of packing and project scheduling problems. Extra details between 2OPP and RCPSP including algorithms are introduced in [29].

One can note that a single resource RCPSP is a special case of 2OPP where items geometric considerations (items cannot be cut) are replaced by resource consumption. Most of the time, depending on the data characteristics, it is possible to compute a 2OPP solution respecting the RCPSP activities starting time. Relaxation of items geometry in the 2OPP results in a resource-constrained project scheduling problem (RCPSP) easier to solve in the sense the problem is less constrained and RCPSP solutions can be investigated in smaller computation time than for packing problems using heuristic and/or metaheuristics. Note that both 2OPP and RCPSP are both NP-Hard to solve.

Since the RCPSP is less constrained than 2OPP, greedy heuristics and meta-heuristic are more efficient to provide quality solutions in reasonable computational time. The 2L-CVRP framework we promote, takes advantages of this feature and it solves a RCPSP-CVRP, i.e. a CVRP with a RCPSP trip check avoiding costly 2OPP trip check. This approach belongs to the GRASP × ELS framework fully described in Section 2.

## 2. GRASP × ELS framework for the 2|UO|L CVRP

### 2.1. Key-features

We propose to solve the 2L-CVRP with unrestricted oriented loading (2|UO|L CVRP) by relaxing packing problem constraints into resource constrained project scheduling problem (RCPSP) constraints. A greedy randomized adaptive search procedure (GRASP) is used to compute high quality RCPSP-CVRP solutions. At the end of the optimization process the RCPSP-CVRP solutions are transformed into 2L-CVRP solutions. During the optimization, trips are checked to be "RCPSP-feasible": items can be loaded into the vehicle with respect to the RCPSP constraints, i.e. at each point of the vehicle length the total width used does not exceed the vehicle width. Note the vehicle width is related to the RCPSP resource availability.

The framework we introduce works in two steps (see Fig. 3): in the first step, the bin-packing constraints are relaxed into RCPSP constraints. The resulting problem to be solved is denoted RCPSP-CVRP since we consider only that a trip must complied with the RCPSP constraints and that no packing constraints hold. Thus the resulting RCPSP-CVRP problem becomes easier to solve. The second step consists in converting a RCPSP-CVRP solution into a 2L-CVRP solution.

The first step is solved by a GRASP × ELS metaheuristic. Solutions are only required to be both "weight-feasible" and "RCPSP-feasible". The load of each vehicle is limited by a coefficient $p$ to limit unsuccessful conversions into a 2L-CVRP solution. However, it is possible that high quality RCPSP-CVRP solutions may not lead to feasible 2L-CVRP solution. Thus, the $nb$ best RCPSP-CVRP solutions built during the GRASP × ELS process are kept. At the end of GRASP × ELS, these solutions are iteratively investigated and the best that can be transformed into a 2L-CVRP solution is kept as the best solution found during the process.

During the optimization, the number of trips can exceed the number of allowed vehicles. These solutions are considered but they are penalized as follows:

$$f(S) = (N(S) - N)\alpha + \sum_{t \in t(S)} f(t)$$

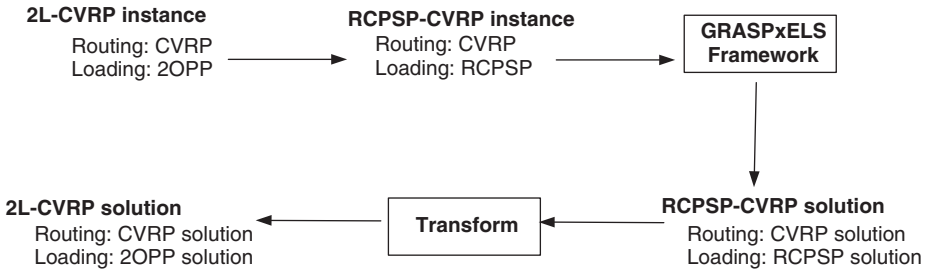


**Fig. 3.** 2L-CVRP optimization flowchart.

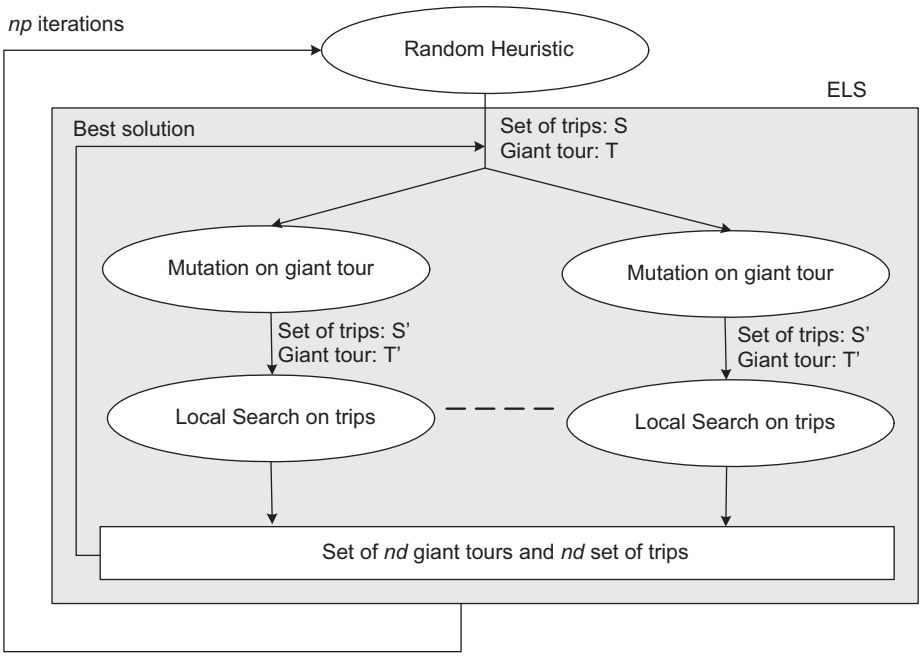


**Fig. 4.** GRASP × ELS principle.

where $N(S)$ is the number of trips in solution $S$, $N$ the number of vehicles, $\alpha$ the penalty, $t(S)$ the set of trips of solution $S$, and $f(t)$ the cost of trip $t$.

### 2.2. GRASP × ELS principle

The purpose of this section is to evoke the principles of GRASP × ELS where:

- greedy randomized adaptive search procedure (GRASP) is a multi-start local search metaheuristic in which each initial solution is constructed using a greedy randomized heuristic.
- evolutionary local search (ELS) is an evolved version of iterated local search (ILS). The purpose of ELS is to better investigate the current local optimum neighborhood, before leaving it whereas the purpose of the GRASP consists in managing diversity in search space investigation.

Starting from an initial solution, each ILS iteration consists in taking a copy of the incumbent solution $S$, applying a perturbation similar to the mutation operator of genetic algorithms, and improving the perturbed solution using a local search. The resulting solution $S'$ becomes the incumbent solution. The evolutionary local search or ELS, introduced in [30] for the routing problems, is similar but, at each iteration $nd$ "children" instead of 1 are generated from $S$, using mutation and local search, and the best child replaces $S$. The framework we promote is a multi-start ELS in which an ELS is applied to the initial solutions generated by greedy randomized heuristics. Such metaheuristic can also be viewed as a GRASP × ELS in which the local search is replaced by an ELS. GRASP × ELS [30,31] is a hybridization of both GRASP and ELS capturing the positive features of both methods (Fig. 4).

### 2.3. Search space investigation strategy

The GRASP × ELS efficiency is based on a swap between solution representations: solutions encoded as giant tours (TSP tours on the $n$ customers) and RCPSP-CVRP solutions encoded as the set of trips (Fig. 5). Such an approach allows GRASP to focus on

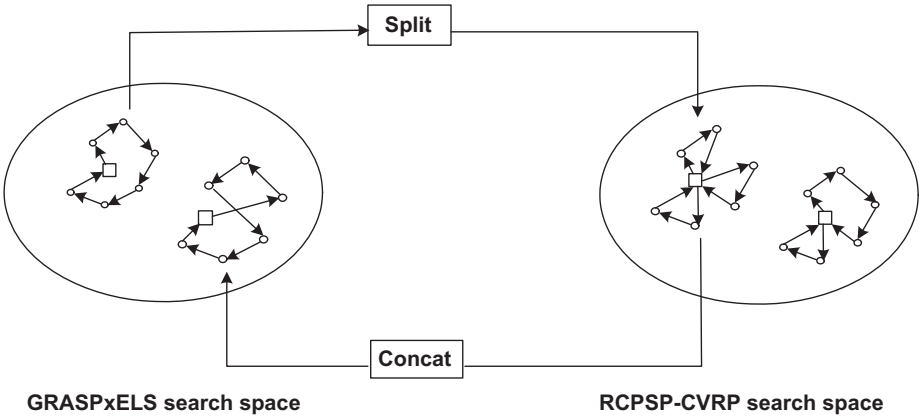


**Fig. 5.** Combination of the two search spaces.

the giant tour space (which is smaller than the space of RCPSP-CVRP solutions) and a giant tour $T$ is converted into an RCPSP-CVRP solution $S$ (with respect to the sequence) using a dedicated splitting procedure (*Split*). Split has been successfully applied to numerous routing problems including the capacitated arc routing problem (CARP [32]), the vehicle routing problem (VRP [33]), the location routing problem (LRP [34,35]). The high quality solutions obtained by Prins [33], alternating between two search spaces (giant tour and VRP solutions) is a first-rate indication of the approach quality. The GRASP × ELS takes advantages in this line of research, by investigating both the space of giant tours and the space of complete 2L-CVRP solutions.

During the GRASP × ELS process, a giant tour $T$ is converted by *Split* into a RCPSP-CVRP solution $S$ with respect to the given sequence and to the RCPSP constraints. The *Concat* procedure converts $S$ into a giant tour $T'$ by concatenating its trips. The giant tour $T'$ can be split again in order to get a new RCPSP-CVRP solution. This process allows alternating between the giant tours space and the RCPSP-CVRP space.

The *Local Search* is a first improvement descent method using several classical VRP neighborhoods to improve the initial RCPSP-CVRP solution: 2-Opt within a trip, 2-Opt between two trips, Swap within a trip and Swap between two trips.

Connecting these components together leads to the GRASP × ELS scheme presented in Algorithm 1. $T$, $S$ and $f(S)$, respectively, denote a giant tour, a RCPSP-CVRP solution and its cost. During the algorithm, the incumbent solution is stored, thus $S^*$ denotes the incumbent solution and $f^*$ is its value. The lines 14–36 correspond to the GRASP × ELS loop of Fig. 4. It generates $np$ pairs ($S$, $T$) used as starting points by the embedded ELS. The $ns$ iterations of ELS are performed in the loop lines 20–35 and the $nd$ parallel mutations are done in lines 22–28. To compute each pair ($S$, $T$), randomized versions of the well-known Clarke and Wright's [36] heuristic and of Path Scanning heuristic [37] are used. The initial solution $S$ is improved by a procedure *Local_Search* introduced in Section 2.6 and it is converted into a giant tour $T$ using the *Concat* procedure. Any giant tour is scanned into trips tackling if a trip $t$ is RCPSP-feasible.

During the GRASP × ELS process, the $nb$ best RCPSP-CVRP solutions are saved into an ordered set $O$ (sorted on increasing costs). At the end of GRASP × ELS, the solutions from $O$ are iteratively inspected and tentatively transformed into a 2L-CVRP solution using a *Transform* procedure (step 41 in Algorithm 1). $O$ is scanned as long as the transformation fails. Thus the scan stops with the best solution that can be transformed into a 2L-CVRP solution. If no solution of $O$ can be successfully transformed, the GRASP × ELS fails and returns $f(S)=\infty$. The *Transform* procedure is detailed in Section 2.9.

In order to be efficient, the GRASP × ELS process must include an intensification procedure to favor convergence into promising search space area for large scale instances. The initial heuristic solutions are discarded if they are worse than the incumbent ($S^*$): this favors the neighborhoods exploration around the best solution. The intensification process is not useful for small and medium scale instances and it is activated using a global GRASP × ELS parameter denoted *intensification*.

### 2.4. Random heuristics to generate initial solutions

Clarke and Wright's heuristic and Golden et al.'s heuristics are used in a wide range of routing problems since they provide high quality solutions in a rather short computational time and since they are easy to randomize. Our heuristic procedure iteratively applies the four considered heuristics: randomized Clarke and Wright (RCW), Path Scanning (PS), randomized Path Scanning (RPS) and a basic random generation (RS). Note that seven criteria are used for the Path Scanning heuristic. All heuristics compute a weight-feasible and load-feasible solution without considering the RCPSP constraints but considering that the area of the vehicle available is limited to $p$ percent of the initial area. Solutions are "load-feasible" and "weight-feasible" only and transform into a giant tour which is lately split into trip using the split procedure (see Section 2.8) which ensures the RCPSP constraints hold for trips, i.e. trips are "RCPSP-feasible". The procedure Generation_of_Initial_Solutions detailed in Algorithm 2 uses the two well-known powerful heuristics denoted Path-Scanning and Clarke and Wright. Clarke and Wright's heuristic introduced in 1964 [36] consists in first providing a number of trips equal to the number of customers to service. Path-Scanning was initially designed for arc routing problems but we have adapted it to the 2L-CVRP.

```
1.  procedure GRASP×ELS
2.  global parameters
3.    np: number of GRASP iterations (initial solutions)
4.    ns: maximum number of iterations per ELS
5.    nr: maximum number of iterations without improvement per ELS
6.    nd: number of diversifications (mutations)
7.    nb: number of high quality solutions saved
8.  output parameters
9.    S*: best 2L-CVRP solution found
10. begin
11.   f* := ∞; O := ∅
12.   for p := 1 to np do
13.     S := call Generation_of_initial_solution ()
14.     T := call Concat (S)
15.     if (f(S) < f*) then f* := f(S); S* := S
16.     else
17.       if (intensification = true) then
18.          S := S*
19.       endif
20.     endif
21.     i, r := 0
22.     while (i < ns) and (r < nr) do    // ELS loop
23.       i := i + 1; f″ := ∞
24.       for j := 1 to nd do             // mutation loop
25.           T′ := call Mutation (T)
26.           S′ := call Split (T′)
27.           S′ := call Local_Search (S′)
28.           T′ := call Concat (S′)
29.           if (f(S′) < f″) then f″ := f(S′); T″ := T′; S″ := S′; endif
30.       endfor
31.       if (f″ ≥ f(S)) then      // check if not improved solution
32.         r := r + 1             // update the number iterations without improvement
33.       endif
34:       if (f″ < f*) then        // if a new best solution
35.         add (O, S)             // add this solution to list O
36.         S*:= S″                // update S*
37.       endif
38.       T := T″;                 // best ELS solution becomes the new initial solution
39.     endwhile
40.   endfor
41.   Iterative check of solutions in O looking for a 2L-CVRP solution
42. end
```

**Algorithm 1.** GRASP × ELS for the 2L-CVRP.

*Path-Scanning* is a greedy heuristic assigning new customers in trips according to a function which depends on five criteria (see [37]). Initially, the heuristic uses a nearest neighbor technique building VRP trips one by one: different rules are used to break ties. In the present implementation, the minimal distance is not used and some non-promising solutions can be obtained without damage for the global metaheuristic performance since these solutions are used for the evolutionary local search scheme which consists in generation of *nd* children. This approach provides a great diversity in the initial solutions generation. Five executions of the heuristic over the five criteria permit to keep the best solution. The five criteria first introduced in [37] are the following: the customer with the maximal distance to the depot (C1); the customer with the minimal distance to the depot (C2); the customer with the maximal ratio between the quantity to deliver and the distance (C3); the customer with the minimal ratio between the quantity to deliver and the distance (C4); the first criterion until the vehicle is half-loaded and the second criterion otherwise (C5). Our procedure includes a randomized version of Clarke and Wright, the deterministic adaptation of path-scanning, a randomized adaptation of path-scanning and also a full randomized generation. One run consists in calling one of these procedures.

Seven criteria are used as input parameter for the two adaptations of path-scanning:

- customer with the maximal distance to the depot (C1′),
- customer with the minimal distance to the depot (C2′);
- customer with the minimal weight to deliver (C3′);
- customer with the maximal weight to deliver (C4′);
- customer with the minimal area to deliver (C5);
- customer with the maximal area to deliver (C6);
- customer with the maximal distance to the depot if the vehicle area is half-loaded and customer with the minimal distance if not (C7′).

Since there is no guarantee on the number of vehicles used by each heuristic, the solution is checked at each iteration and stops as soon as the number of vehicles satisfies the number of available vehicles with a number of iterations upper bounded by *ne*. If a non-feasible solution is obtained (loop of Algorithm 2 stops because $i > ne$), a penalty is included according to the penalty function defined in Section 2.1. At the end of the procedure, trips of the solution are concatenated to get a giant tour. This giant tour is then submitted to Split to obtain RCPSP-feasible trips. One call to *Generation_of_initial_solution* (Algorithm 2) leads to the execution of one heuristic: two times for both *RCW* and *RS*, one time with the seven criteria for *PS* and *RPS*. Note that current is a global parameter to provide a loop over the heuristics.

```
1.  procedure Generation_of_initial_solution
2.  input parameters
3.    ne: maximal number of iterations
2.  output parameters
3.    S: a RCPSP-CVRP solution
4.  begin
5.    current := 1;  cpt := 0;    i := 1
6.    while ( (i≤ne) and (no solution found) ) do
7.      case current of
8.        case 1:
9.          S := call RCW ()
10.         cpt := cpt + 1
11.         if (cpt > 2) then
12.           cpt := 0; current := current + 1
13.         endif
14.       case 2 :
15.         S := call PS (criterion)
16.         criterion := criterion + 1
17.         if (criterion > 7) then
18.           criterion := 1; current := current + 1
19.         endif
20.       case 3 :
21.         S := call RPS (criterion)
22.         cpt := cpt + 1
23.         if (cpt > 2) then
24.           cpt := 0; criterion := criterion + 1
25.           if (criterion > 7) then
26.             criterion := 1; current := current + 1
27.           endif
28.         endif
29.       case 4 :
30.         S := call RS ()
31.         cpt := cpt + 1
32.         if (cpt > 2) then
33.           cpt := 0; current := 1
34.         endif
35.     endcase
36.     i:=i+1
37.     T := call Concat (S); S := call Split (T);
37.   endwhile
38.   return S
38. end
```

**Algorithm 2.** Generation of initial solutions.

The procedure *Generation_of_initial_solution* generates a lot of different solutions by taking advantage of the four procedures and of the seven criteria introduced in the two versions of Path-Scanning.

### 2.5. Mutation

The mutation operator is defined on giant tours $T = (T_0, T_1, \ldots, T_{n(T)})$ (where $T_i$ is the $i$th trip and $n(T)$ is the number of trips in $T$) and based first on generation of new trip concatenation order and second on customer exchange to obtain a children giant trip $T'$.

During the first step, one cutting trip $T_i$ is randomly selected in $T$ and the substring $T_i,\ldots,T_{n(T)}$ is copied into $T'_1,\ldots,T'_{n(T)-i+1}$. Finally, $T$ is swept from 1 to $T_{t-1}$ to complete $T'$ with the missing trips. Second two customers are randomly chosen in $T'$ and exchanged in $T'$.

### 2.6. Local search

The purpose of the local search is to improve a RCPSP-feasible solution by investigating the RCPSP-CVRP space. As mentioned before, the procedure relies on classical 2-Opt and Swap moves. These moves exist in two versions: inside a single trip and between two trips. At

each iteration the local search uses the procedures *LS_2Opt_Intra, LS_2Opt_Inter*, *LS_Swap_Intra, LS_Swap_Inter*. These procedures define an iterative search scheme using, respectively, the following neighborhood structures 2-Opt inside a trip, 2-Opt between two trips, Swap inside a trip and Swap between two trips. The order in which these procedures are called depends on probability $q$ as stressed in Algorithm 3.

This algorithm depends on a threshold $\varepsilon$ used to guarantee a minimal improvement on the solution cost. The iterative search stops when the improvement is lower than $\varepsilon$ (typically $\varepsilon=1$) or when the maximal of iterations has been reached. The first improvement criterion is used when exploring the neighborhood and the *Check_RCPSP* procedure is called on each move. An additional neighborhood (*LS_Split*) is used when the current solution has not been improved by at least one of the first four local searches. It consists in random trips concatenations into giant tours and projecting it back into the RCPSP-CVRP solution space using *Split*. Its purpose is to check if *Split* is able to find a better solution with the modified sequence. As *LS_Split* requires *Split*, its time complexity is high nevertheless it provides an efficient local search scheme. Thus it is called only if the current solution has not been sufficiently improved.

Compared to more elaborate neighborhood structures, we choose to restrict the local search to the 2-Opt and to the Swap for two reasons: the time complexity remains low and the intrinsic structure of the GRASP × ELS scheme handles well a restricted local search.

```
1.  procedure Local_Search
2.  input parameters
3.    S:        RCPSP-feasible initial solution
4.  global parameters
5.    max_iter: maximum number of iterations
6.    ε       : threshold on the improvement
7.    q       : probability
8.  output parameters
9.    S: incumbent RCPSP-feasible solution
10.  begin
11.   iter := 0
12.   do
13.     old_val := f(S)
14.     S := call LS_2Opt_Intra (S)
15.     S := call LS_2Opt_Inter (S)
16.     if (random < q) then      // random : random number generation between 0..1
17.        S := call LS_Swap_Intra (S)
18.        S := call LS_Swap_Inter (S)
19.     else
20.        S := call LS_Swap_Inter (S)
21.        S := call LS_Swap_Intra (S)
22.     endif
23.     if (old_val- f(S) ≤ ε) then S := call LS_Split (S) endif
24.     iter := iter + 1
25.   while (f(S) + ε < old_val) and (iter < max_iter)
26.   return S
27. end
```

**Algorithm 3.** Local search.

### 2.7. A simple and effective RCPSP resolution (check_RCPSP)

During the GRASP × ELS, a RCPSP with a single resource and with no precedence constraints is addressed. The vehicle length gives an upper bound of the completion time of the last activity to schedule. For each trip, we have to solve a RCPSP where activity duration and requirement in resource are given by item length and width, respectively. A trip is RCPSP-feasible if the makespan of the corresponding RCPSP does not exceed the vehicle length. Each time a trip is modified in the local search or each time the Split propagates labels by adding a new trip, a check RCPSP-feasibility must be achieved. Thus the efficiency, in terms of quality and in terms of speed, of the corresponding algorithm is critical.

The schedule generation schemes (SGS) [38] for RCPSP are based on priority-rule relying both the precedence constraints and the limitation on the total amount of resources. Initially, no activity is scheduled. At each iteration, activity with the highest priority is selected and scheduled. The priority of each activity depends on both the precedence constraints and the resources consumption and previously scheduled activities.

The efficiency of priority rules depends on the number of resources and on the number of precedence constraints to generate adequate activity list. According to Kolisch [38] WCS is one of the best priority rules. It provides an average deviation of 3.71% to optimality on Patterson instances with a short computational time. However, since there is no precedence constraint and only one resource is used, the priority rule computation is unattractive and it does not induce a profitable activity list. Since our RCPSP handles few constraints, it leads to activities with identical (or similar) priorities. One could use a randomized-WCS (random selection of activity between the best activities) which required time consuming priorities computation without any profitable result since WCS is not adequate for this specific RCPSP (one resource only with no precedence constraint). Thus, we propose a randomized activity selection which has been benchmarked and which seems to be highly efficient on this specific RCPSP.

Our SGS (Algorithm 4) performs $n$ iterations, one per activity. At an iteration $k$, the partial schedule contains $k$ activities. Several variables are associated to iteration $k$:

- $t_k$: the schedule time (current time);
- $A_k$: the active set, i.e. the activities scheduled but not finished at time $t_k$;
- $R_k$: the amount of resources available at time $t_k$, $R_k = M - \sum_{i \in A_k} r_i$ where $M$ is the total amount of resource available (the vehicle width) and $r_i$ is the requirement in resource of activity $i$ (the item width);
- $D_k$: the decision set, i.e. the activities which can be scheduled at $t_k$ (one activity $i$ can be scheduled at $t_k$ if all its predecessors are finished and if $r_i \leq R_k$).

```
1. procedure Randomized_SGS
2. input parameters
3.   set: set of na activities j of duration dj
4.   M  : maximal project duration
5. output parameters
6.   ESj: earliest starting time of activity j
7.   LSj: latest starting time of activity j
10. begin
11.  t0 := 0
12.  D0 := set          // set of activities which can be scheduled at tk = 0
13.  A0 := {}
14.  for k := 1 to na do
15.     randomly choose an activity j in Dk
16.     schedule j at tk: ESj := tk
17.     update Ak, Dk
18.     while (Dk is empty)
19.       tk := min{EFj / j in Ak}
20.       update Dk, Ak
21.     endwhile
22.   endfor
23.  if max(ESj+dj)<M then
24.    compute LSj
25.  endif
26. end
```

**Algorithm 4.** A randomized schedule generation scheme for the 2L-CVRP.

Note the procedure *Randomized_SGS* looks for a RCPSP solution less or equal than a maximal project duration $M$. If the makespan (earliest finish time of the last activity) is less or equal than $M$ the latest starting time of activities are computed and the procedure return both $ES_j$ and $LS_j$ for each activity $j$. To check the RCPSP-feasibility, the basic procedure *Basic_Check_RCPSP* (Algorithm 5) calls the *Randomized_SGS* procedure until a RCPSP solution is feasible (all activities end before the maximal project duration) or until the maximal number of iterations is reached.

```
1.  procedure Basic_Check_RCPSP
2.  input parameters
3.    set: set of activities j of duration dj
4.    M  : maximal project duration
7.    nm : maximal number of attempts
8.  output parameters
9.    res     : result for the procedure (success/failure)
10.   makespan: the makespan
10. begin
11.   k := 1
12.   do
13.     (ES,LS) := call Randomized_SGS (set, M)
14.     k := k + 1
15.   while (max{ESj+dj} > M) and (k ≤ nm)
16.   res := (max{ESj+dj} ≤ M) ; makespan := max{ESj+dj}
17. end
```

**Algorithm 5.** Basic RCPSP check.

The aim of the procedure *Check_RCPSP* is to check if the makespan of the RCPSP does not exceed the vehicle length $L$. Trying to speed up the Check_RCPSP procedure, we propose to iteratively use the *Basic_Check_RCPSP* procedure with different decreasing total project duration. The main idea consists in considering that if the best solution found over the first iterations is strongly greater than the vehicle length, then it is possible to state that extra investigation are useless for RCPSP resolution.

The procedure *Check_RCPSP* uses two arrays $\Psi$ and $\Delta$ of length $K$ where $K$ is the number of steps investigated in *Check_RCPSP*. $\Psi$ keeps the successive value (in percent) and $\Delta$ is an array of iterations assigned to each percent in $\Psi$. For example let us consider $L=40$, $\Psi=(1.25;1.15;1.10;1.00)$ and $\Delta=(10;100;300;1000)$. Thus, the first call to *Basic_Check_RCPSP* is achieved with $40\times1.25=50$ for the maximal project duration and 10 iterations. If in 10 iterations the best solution found is greater than 50, the algorithm stop assuming the probability of finding a makespan less than $L=40$ is too low. Otherwise, starting from this solution, 100 iterations are used to find a schedule with a duration less than $40\times1.15=46$.

```
1.  procedure Check_RCPSP
2.  input parameters
3.    set: set of activities j of duration d_j
6.  global parameter
7.     L  : vehicle length
8.    Ψ [] : array of percent
9.    Δ [] : array of number of iterations
10.  output parameters
11.    res: result for the procedure (success/failure)
12.  begin
13.   k := 1
14.   do
15.     (res, makespan) := call Basic_Check_RCPSP (set, L*Ψ [k], Δ [k])
16.      k := k + 1
17.   while ( (res = true) and (makespan>M) and (k<K))
18. end
```

**Algorithm 6.** RCPSP check.

During the experiments, we noted that *Check_RCPSP* usually works better with a larger amount of resources than the duration. Thus, as the vehicle length is larger than its width, the two dimensions are exchanged before entering the procedure.

### 2.8. Split procedure

As previously stressed, Split is a key-procedure to convert a giant tour into a RCPSP-CVRP solution (with respect to the sequence). It is based on the classical Split procedure [39,32,33], tuned to address the specific RCPSP-CVRP constraints.

The Split procedure first builds an auxiliary digraph $H_T=(X, Y, Z)$ where $X$ is a set of $n+1$ nodes indexed from 0 to $n$. Node 0 is a dummy node, while the nodes $1,\ldots,n$ correspond to the sequence of the giant tour $T=(v_1,\ldots,v_n)$. An arc $(i, j)$ belongs to $Y$ if a trip servicing customers $v_{i+1}$ to $v_j$ (included) is both weight-feasible and RCPSP-feasible. The weight of the arc $(i; j)\in Y$ is the trip cost

$$z_{ij}=c_{0v_i}+\sum_{k=i,\ldots,j-1}c_{v_kv_{k+1}}+c_{v_j0}$$

Optimally splitting $T$ (Fig. 6) corresponds to a min-cost path from node 0 to node $n$ in $H$. An initial label is set at node 0. The labels are propagated from node to node in $H$ using the arcs and the best label at node $n$ is kept.

Let $L_i^p=(N_i^p,z_i^p,k,j)$ be the $p$th label assigned to node $i$. It corresponds to a feasible split of the initial customers $t_1,\ldots,t_i$ into trips. $N_i^p$ is the number of vehicles remaining available, $z_i^p$ is the cost of the trips previously built and $(k, j)$ is the reference to its father label, e.g. $L_j^k$, the $k$th label at node $j$. The initial label at node 0 is defined as $L_0^1=(N,0,-1,-1)$. It corresponds to the empty solution where all the

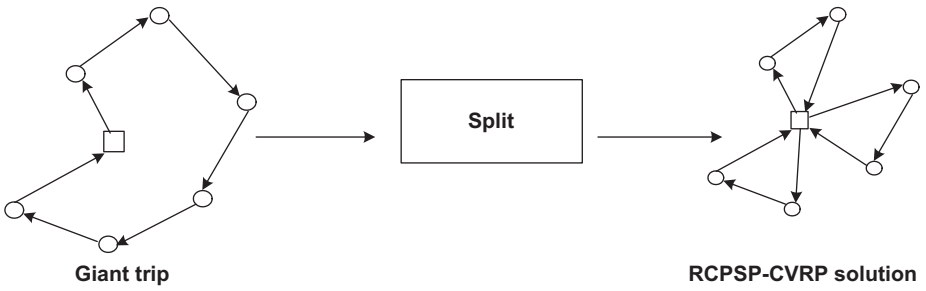


**Fig. 6.** Split transformation.

vehicles are available. Given the arc $(i, j) \in A$, label $L_i^p$ generates label $L_j^q = (N_j^q, z_j^q, i, p)$ by propagation as follows:

- $N_j^q = N_i^p - 1$
- $z_j^q = z_i^p + z_i$

Since large number of labels on node has significant impacts the efficiency, dominance rules can be defined considering that label $L_i^p$ dominates $L_i^q$ if one of the following conditions holds:

$$\begin{cases} (N_i^p > N_i^q) \text{ and } (z_i^p \le z_i^q) \\ (z_i^p < z_i^q) \text{ and } (N_i^p \ge N_i^q) \end{cases}$$

The critical path leading to the best final label defines the trips of the solution. The procedure Split is detailed in Algorithm 7. For each node $i$, $NB[i]$ gives the number of related labels. The procedure *Check_Domination_On_Node* checks if the new label $L$ is dominated by another label at node $j$. The procedure Insert inserts this label into the set of node $j$ labels and removes the dominated labels. The number of labels is updates accordingly.

```
1.  procedure Split
2.  input parameters
3.    T: giant tour
4.  output parameters
5.    S: RCPSP-CVRP solution
6.  global parameter
7.    D      : maximal vehicle weight capacity
8.    d_i    : total items weight of customer i
9.    c_ij   : cost from customer i to j
10.   n      : number of customers
11.  begin
12.    L_0^1 := (N,0,-1,-1),  S := ∅
13.    for i := 1 to n do L_i := ∅ endfor
14.    for i := 0 to n - 1 do
15.     j := i + 1
16.     trip := ∅; client := ∅
17.     repeat
18.       prev := client
19.       client := T_j
20.       trip := trip + client
21.       if (j = i + 1) then
22.         trip_load := d_client
23.         trip_cost := c_depot,client + c_client,depot
24.       else
25.         trip_load := trip_load + d_client
26.         trip_cost := trip_cost +c_prev,client +c_client,depot -c_prev,depot
27.       endif
28.       check := (trip_load < D) and (Check_RCPSP(trip) = true)
29.       if (check = true) then
30.         for p := 1 to NB_i do
31.           let L_i^P := (N_i^p, z_i^p, k, j) be the current label
32.           propagate on j: L := (N_i^p - 1, z_i^p + z_ij, i, p)
33.           if (Check_Domination_On_Node( L , j, NB_j) = false) then
34.             call Insert(L, j, NB_j)
35.           endif
36.         endfor
37.       endif
38.       j := j + 1
39.     until (check = false) or (j > n)
40.   endfor
41.   if (NB_n > 0) then
42.     S := call extract_trips ()
43.   endif
44. end
```

**Algorithm 7.** Split.

### 2.9. Transformation of a RCPSP-CVRP solution into a 2L-CVRP solution

Several authors, including Iori et al. [10], have previously noticed that greedy algorithms could lead to efficient frameworks by carefully managing the classical envelope (or contour) as defined in Martello et al. [40] and addressing only normal fillings. Exact methods based on linear formulations cannot be used due to excessive computational time. As a consequences of the framework we introduce, the transformation of a RCPSP-CVRP solution into a 2L-CVRP solution is only done at the end of GRASP × ELS. In order to be efficient, the transformation has to take advantage of the output of the RCPSP check. In a RCPSP-CVRP solution, items are associated to an abscissa. Note that only one bin is available (the vehicle) and that the $x$-position $x_i$ of an item $i$ can be given either by its earliest starting time ($ES_i$) or by its latest starting time ($LS_i$) computed by the procedure *Check_RCPSP*. A 2L-CVRP solution is first investigated with the earliest starting time ($ES_i$) and then (if required) using the latest starting time ($LS_i$).

The Algorithm *RCPSP_To_Packing* (see Algorithm 8) investigates iteratively all the $y$-positions starting from 0 until all items are packed or until the remaining items cannot be packed (we have reached the vehicle width). $D_y$ is defined as the set of items $i$ which can be scheduled at position $(x_i, y)$. The envelope of items already packed is given by the function *Front*($x$), which returns the smallest $y$ available at abscissa $x$. At each iteration, *min_front is* the smallest value of the front where remaining items can be scheduled, is considered. Then a random selection is done in $D_{min\_front}$. The item is scheduled and the envelope is updated. The outer loop (steps 11–24) tries to pack the maximal number of items. When $D_y$ is empty, the next $y$-position is investigated with respect to the minimal front value where items can be scheduled.

```
1.  procedure RCPSP_To_Packing
2.  input parameters
3.    set: set of items of length l_j and of width w_j
4.    x_i: abscissa of item i
5.  output parameters
6.    res: result of the procedure (success/failure)
4.    y_i: ordinate of item i
5.  global parameters
6.    L: vehicle length
7.    W: vehicle width
7.  begin
8.    A := set
9.    y := 0
10.   for x := 0 to L do front[x] := 0 endfor
11.   repeat
12.     D_y := {items which can be scheduled at y according to x_i}
13.     if (D_y = ∅) then
14.       prec_y := y
15.       y := min {front(x) | front(x) > y}
16.     else
17.       randomly choose i in D
18.       A := A \ {i}
19.       schedule i at (x_i, y); y_i := y
20.       for x := x_i to x_i + l_i -1 do
21.         front(x) = front(x) + w_i
22.       endfor
23.     endif
24.   until (A = ∅) or (y ≥ W)
25.   res := (A = ∅)
26. end
```

**Algorithm 8.** RCPCP_To_Packing.

The following example illustrates the algorithm. Table 6 contains the items to be packed into a vehicle of size 35 × 20.

**Table 6**
Items to be packed.

| | Width | Length | $x_i = ES_i$ |
|---|---|---|---|
| *a* | 6 | 18 | 0 |
| *b* | 12 | 5 | 0 |
| *c* | 8 | 5 | 5 |
| *d* | 10 | 10 | 10 |
| *e* | 5 | 7 | 18 |
| *f* | 5 | 15 | 20 |

At the beginning, no items are scheduled as shown in Fig. 7. Initially, all the items can be scheduled at $y=0$, thus $D=A$. Item $b$ is randomly chosen and scheduled. The front and $D$ are updated: $D=\{c, d, e, f\}$ as item $a$ cannot be scheduled at $y=0$ anymore. The next item is randomly chosen, say $c$.

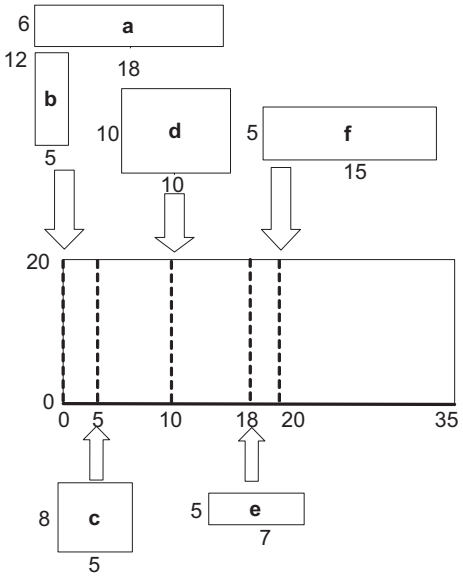


**Fig. 7.** Initial state.

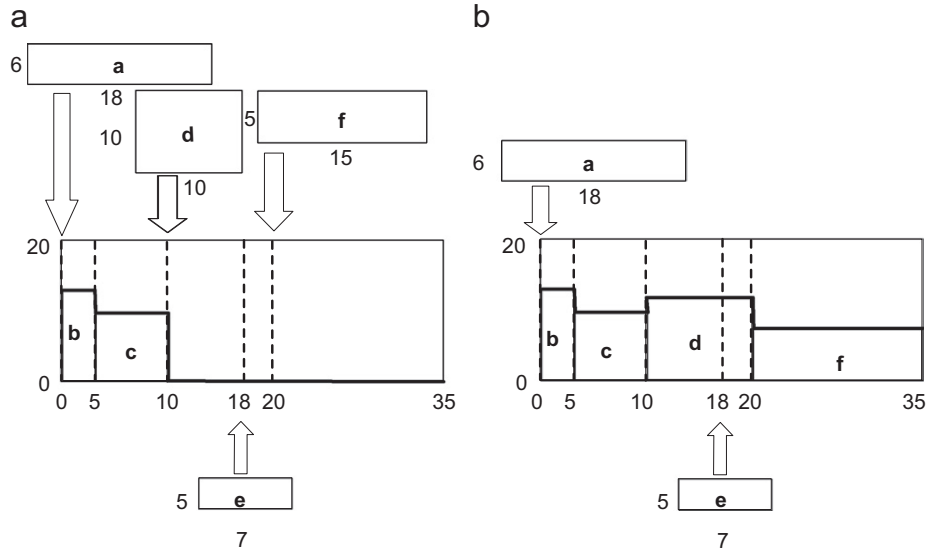


**Fig. 8.** Packing the items: (a) after scheduling $b$ and $c$ and (b) after scheduling $b$, $c$, $d$, $f$.

**Table 7**
Parameter settings for class 1.

| | Small scale | Medium scale | Large scale |
|---|---|---|---|
| *np* | 20+[*n*/10]∗2 | 20+[*n*/10]∗20 | Infinite |
| $\alpha$ | 100 000 | 100 000 | 100 000 |
| *ne* | 5 | 5 | 5 |
| *ns* | 20 | 20 | 20 |
| *nr* | 15 | 15 | 15 |
| *nd* | 10 | 15 | 15 |
| *nb* | 20 000 | 20 000 | 20 000 |
| *max_iter* | 50+[*n*/10] | 50+[*n*/10] | 50+[*n*/10]∗7 |
| *p* | 1 | 1 | 1 |
| $\varepsilon$ | 1 | 1 | 1 |
| *Intensification* | No | No | Yes |

The front and $D$ are updated: $D=\{d, e, f\}$ (see Fig. 8). Suppose the next item is $d$. $D=\{f\}$ as item $e$ cannot be scheduled at $y=0$ anymore. The item $f$ is scheduled (see Fig. 8.b). So, the minimal value of the front where a remaining item can be scheduled is $y=10$ and $D=\{e\}$. Item $e$ is scheduled, $y=12$ and $D=\{a\}$. Item $a$ is scheduled, $D=\{\ \}$ and a packing solution has been found.

Note that computing a feasible packing solution depends on the date $x_i$ assigned to the items. The procedure *Transform* performs several successive calls to the procedure *RCPSP_To_Packing* until a feasible packing is found. At each iteration, a new schedule is generated. A transformation is experienced first using the earliest starting times $ES_i$ as $x_i$ then using the latest starting times $LS_i$.

```
1.  procedure Transform
2.  input parameters
3.    S: solution (set of trips, where S_i is the i^th trip)
4.  global parameters
5.    iter_max: maximal number of iterations per trip
6.  output parameters
7.    x_j: starting time of activity j
8.    y_j: y-position for activity j
9.    res: result of the procedure (success/failure)
10. begin
11.   i := 1, res := true
12.   while (i ≤ number of trips in S) and (res = true) do
13.     k := 1; res_trip := false
14.     while (res_trip = false) and (k ≤ iter_max) do
15.       call Randomized_WCS (S_i)
16.       for l := 1 to ne do
17.         (res_trip, y) := call RCPSP_To_Packing (S_i,ES)
18.         if (res_trip = true) then x_j = ES_j; break endif
19.       endfor
20.       if (res_trip = false) then
21.         for l := 1 to ne do
22.           (res_trip, y):= call RCPSP_To_Packing (S_i,LS)
23.           if (res_trip = true) then x_j = LS_j; break endif
24.         endfor
25.       endif
26.       k := k + 1
27.     endwhile
28.     res := res_trip
29.     i := i + 1
30.   endwhile
31. end
```

**Algorithm 9.** Transform.

### 2.10. Hash function for RCPSP

Since a huge number trips are evaluated during the GRASP × ELS and because each trip required a RCPSP-feasibility check the process (which checks the RCPSP feasibility) must be time efficient. It can take advantages of both a map to store the result of the RCPSP-feasibility for trips previously evaluated and of a hash function to address the results within the map. The hash function $h(t)$ associated to a trip $t$ is defined as follows:

$$h(t) = \prod_{i \in t} prime[i] \bmod K$$

where $prime[i]$ is the $i$th prime number and $K$ is a constant. We consider a table of ordered prime numbers (2, 3, 5, 7, …). The value of $K$ impacts the size of the map and the probability of collision (so $K$ must be as large as possible considering the memory available). Two different trips $t$ and $t'$ leading to the same hash value are stated to be in collision. This hash function has interesting properties on RCPSP-equivalent trips.

**Definition.** Two trips $t$ and $t'$ which visit the same set of customers are RCPSP-equivalent.

As the set of customers is the same for $t$ and $t'$, $t$ is RCPSP-feasible if and only if $t'$ is RCPSP-feasible. Thus the following property holds:

**Property.** *RCPSP-equivalent trips $t$ and $t'$ have the same hash value ($h(t)=h(t')$).*

This property is interesting as it means that only one RCPSP-feasibility check has to be done for all the trips having the same set of customers. The value stored in the map, along with the trip, is defined as follows:

$$F\left(h(t)\right) = \begin{cases} 0 & \text{if RCPSP-feasibility has not yet been done} \\ 1 & \text{if } t \text{ is RCPSP-feasible} \\ -1 & \text{if } t \text{ is not RCPSP-feasible} \end{cases}$$

The improved version of *Check_RCPCP* (see Algorithm 10) takes into account the hash function and the map. The structure map stores the trips already checked and the structure F stores the result of the RCPCP check. To further improve the speed of the procedure, only trips requiring more than a given number of iterations in *Check_RCPSP* are considered for insertion in the map. Such strategy aims at limiting collisions by giving priority, in the storage, to the RCPSP solution whose check has been time consuming.

```
1.  procedure improved_check_RCPSP
2.  input parameters
3.    t  : trip
4.    set: set of activities of trip t
5.  output parameters
6.    res: result of the procedure (success/failure)
7.  begin
8.    if (F(h(t)) = 0) then           // set was never checked
9.      res := call check_RCPSP(set)
10.      if (the number of iterations used in Check_RCPSP is large enough) then
11.       map(h(t)) := t
12.      if (res = true) then
13.        F(h(t)) := 1
14.      else
15.        F(h(t)) := -1
16.      endif
17.    endif
18.  else                            // set has already been checked
19.    if (t = map(h(t))) then       // RCPSP-equivalent
20.      res := (F(h(set)) = 1)
21.    else                          // not RCPSP-equivalent
22.      res := call check_RCPSP(set)
23.    endif
24.  endif
25. end
```

**Algorithm 10.** Improved RCPCP check.

## 3. Computational evaluation

### 3.1. Implementation and benchmarks used

We report results on the set of instances used in previous publications [7–9]. Each instance is divided into five classes: class 1 defines a single $1 \times 1$ item for each customer corresponding to basic CVRP instances (there is no 2OPP to solve). Classes 2–5 contain instances with non-unit item sizes. The item size has been randomly generated to define "vertical" instances (width greater than length),

**Table 8**
Parameter settings for classes 2–5.

| | Small scale | Medium scale | Large scale |
|---|---|---|---|
| *np* | 20+[*n*/10] | 20+[*n*/10] | 20+[*n*/10] |
| $\alpha$ | 100 000 | 100 000 | 100 000 |
| *ne* | 5 | 5 | 5 |
| *ns* | 20 | 20 | 20 |
| *nr* | 15 | 15 | 15 |
| *nd* | 10 | 15 | 15 |
| *nb* | 20 000 | 20 000 | 20 000 |
| *max_iter* | 50+[*n*/10] | 50+[*n*/10] | 50+[*n*/10] |
| *p* | 0.95 | 0.95 | 0.95 |
| $\varepsilon$ | 1 | 1 | 1 |
| *K* | 1 000 000 | 1 000 000 | 1 000 000 |
| *Intensification* | No | No | Yes |

**Table 9**
Comparative performance of processors.

| | Gendreau et al. [7] | Zachariadis et al. [8] | Fuellerer et al.[9] | GRASP × ELS |
|---|---|---|---|---|
| Computer | PIV 1.7 GHz | PIV 2.4 GHz | PIV 3.2 GHz | Opteron 2.1 GHz |
| OS | ? | Windows XP | Linux | Linux |
| Language | C | Visual C++ | C++ | C++ |
| Speed factor | | | 1 | 0.66 |
| Time limit | 1 h | 1 h | 1 h | 1 h 30 |
| No. of runs | 1 | 200 | 10 | 10 |

**Table 10**
packing algorithms on several packing problems.

| No. of items | Area | CPLEX | | Branch and bound | | RCPSP_To_Packing | |
|---|---|---|---|---|---|---|---|
| | | Answer | CPU(s) | Answer | CPU(s) | Answer | CPU(s) |
| 10 | 692 | True | 0.58 | True | 0.14 | True | < 0.01 |
| 11 | 673 | True | 0.23 | True | 0.39 | True | 0.11 |
| 11 | 736 | False | 1362.66 | False | 78.11 | False | 2.86 |
| 12 | 702 | True | 1.06 | True | 2.22 | True | < 0.01 |
| 12 | 737 | True | 264.11 | True | 11.23 | True | < 0.01 |
| 12 | 737 | True | 37.20 | True | 2500.66 | True | 0.02 |
| 13 | 664 | True | 0.30 | True | 0.23 | True | 0.02 |
| 13 | 759 | – | 10800.00 | – | 10800.00 | True | 0.40 |
| 15 | 715 | True | 3.59 | True | 215.88 | True | < 0.01 |
| 15 | 727 | True | 8.88 | True | 1244.66 | True | < 0.01 |
| 16 | 726 | True | 67.28 | True | 3357.28 | True | 0.02 |
| 17 | 488 | True | 0.50 | True | 1.34 | True | 0.02 |
| 17 | 746 | True | 313.97 | – | 10800.00 | True | 0.28 |
| 18 | 639 | True | 0.92 | True | 2.01 | True | 0.34 |
| 18 | 670 | True | 1.50 | True | 2.80 | True | < 0.01 |
| 18 | 770 | True | 65.30 | – | 10800.00 | True | 0.67 |
| 19 | 739 | True | 1315.73 | True | 40.63 | True | 0.02 |
| 21 | 715 | – | 10800.00 | – | 10800.00 | True | < 0.01 |
| 21 | 723 | – | 10800.00 | True | 0.45 | True | 0.02 |
| 25 | 773 | – | 10800.00 | – | 10800.00 | True | 1.56 |

"homogenous" instances and "horizontal" instances. The number of items for each customer for class $i$ has been generated according to an uniform distribution in the interval $[1;i]$ (see [7] for more details about the instances characteristics). These instances can be downloaded at http://www.or.deis.unibo.it/research.html. As mentioned at the beginning the best results published so far for those instances use Ant Colony [9]. The details of the solutions are available at http://prolog.univie.ac.at/research/VRPandBPP/. We also report the results from [7,8]. The solutions produced by GRASP × ELS are available at http://www.isima.fr/~lacomme/2lcvrp/2lcvrp.html. All procedures in our framework are implemented in C++ using the g++ compiler. Numerical experiments were carried out on a 2.1 GHz computer running Linux. Since the GRASP × ELS is a random search algorithm, each instance was solved 10 times and the best found solution over the runs is kept with the CPU time required to reach this value. The set of parameters used, for small scale (from 01 to 15), medium scale (from 16 to 19) and large scale (from 20 to 36) instances, is given in Tables 7 and 8. For all instances, $\Psi=(1.25;1.15;1.10;1.00)$ and $\Delta=(10;100;300;1000)$.

To provide a fair comparative study, the computational time of each method has been scaled by the speed factor presented in Table 9. This coefficient takes into account the MIPS performance of each processor.

All previously published methods were benchmarked over 1 h of computational time, i.e. 1 h of computation is assigned for one run of the method. Since the reference results [9] have been obtained on a computer 1.5 times faster, the GRASP × ELS time limit is set to 1h30. As stated in Table 9 the number of runs used in methods varies from 1 to 200 which does not favor fair comparative study. Note also that all authors report the best found solution using the total number of runs reported in Table 8.

### 3.2. Performance of the procedure RCPSP_To_Packing

To solve the packing problem, our heuristic uses the RCPSP solution. In order to assess its performance, we compare it with two efficient exact methods. The first one is a MIP formulation proposed by Pisinger and Sigurd [14] solved with Cplex 11. The second one is a branch and bound developed by Martello et al. [41]. It is initially dedicated to the 3D bin packing problem and looks for the minimal number of bins. Several trips coming from several instances are evaluated using the three packing methods. The area of the vehicle is set to 800. In Table 10, the first two columns refer to the number of items and to the total area required by the items. Then, the results from the MIP formulation (column CPLEX), the branch and bound and our method are presented. For each method, the answer (packable or not packable) along with the CPU time in seconds is reported. A time limit of 3 h has been set. As solutions are already RCPSP-feasible, few answers might be 'false'. Only one such situation happens and it requires much more time, comparatively to problems of equivalent size. For the 'true' answers, the time depends on the number of items and on the area required by the items. The approach using MIP formulation seems to be slightly better than the branch and bound. However, both are dominated by our approach in terms of CPU time on these instances. Note that our approach is heuristic: there is no guaranty on the answer, even though no erroneous answer has been reported in our experiments.

**Table 11**
Results for class 1 instances.

| | Gendreau et al. [7] | Zachariadis et al. [8] | Fuellerer et al. [9] | GRASP × ELS |
|---|---|---|---|---|
| Nb best | 18 | 21 | 23 | **34** |
| Nb record | 0 | 0 | 2 | **10** |
| Avg. value | 792.31 | 777.75 | 776.04 | **770.77** |

**Table 12**
Results on class 2–5 instances.

| | | Gendreau et al. [7] | Zachariadis et al. [8] | Fuellerer et al. [9] | GRASP × ELS |
|---|---|---|---|---|---|
| Class 2 | Nb best | – | 3 | 15 | **32** |
| | Nb record | – | 0 | 4 | **21** |
| | Avg. value | – | 1205.45 | 1150.68 | **1140.44** |
| Class 3 | Nb best | – | 3 | 7 | **36** |
| | Nb record | – | 0 | 0 | **29** |
| | Avg. value | – | 1217.40 | 1174.98 | **1149.14** |
| Class 4 | Nb best | – | 3 | 8 | **36** |
| | Nb record | – | 0 | 0 | **28** |
| | Avg. value | – | 1223.45 | 1191.59 | **1168.25** |
| Class 5 | Nb best | – | 8 | 15 | **32** |
| | Nb record | – | 0 | 4 | **21** |
| | Avg. value | – | 1078.24 | 1059.55 | **1052.29** |
| Average | Nb best | 0 | 0 | 3 | **36** |
| Class 2–5 | Nb record | 0 | 0 | 0 | **33** |
| | Avg. value | 1216.08 | 1181.13 | 1144.20 | **1127.53** |

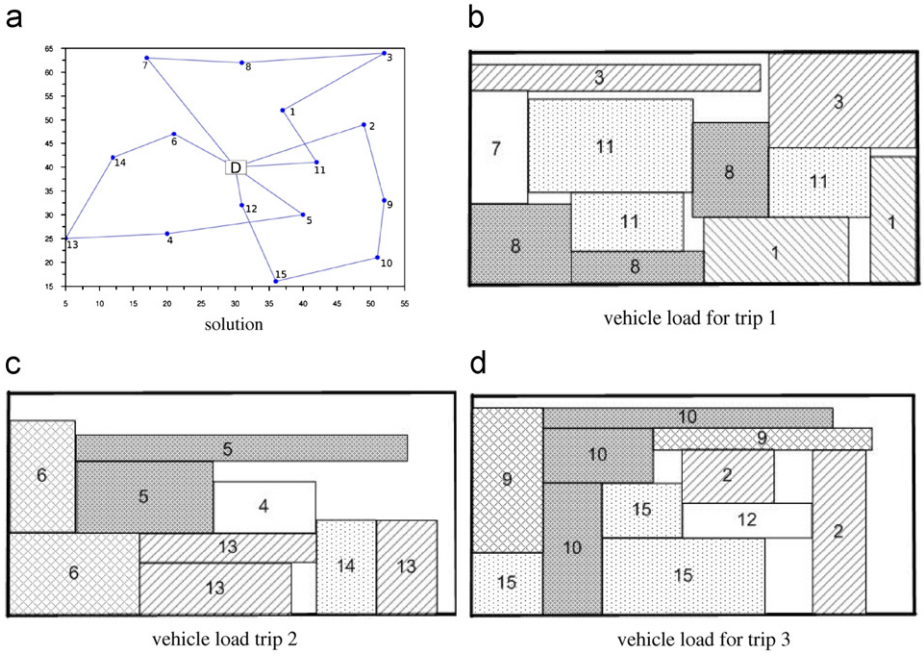


**Fig. 9.** Solution for instance 01 class 3.

### 3.3. Average results on CVRP instances

Class 1 instances are pure CVRP instances as all items are $1 \times 1$ squares. On those 36 instances, the method competes with the best published methods of [7–9] (see Table 11). GRASP × ELS gets the best Result 34 times and provides 10 new best solutions. Details on the results for each instance can be found in Table A1. Row "nb best" gives the number of times a method provides the best result and "nb of record" gives the number of times the method gives a solution strictly better than all others methods.

### 3.4. Average results for 2L-CVRP instances

Average results for classes 2–5 are presented in Table 12. The same information is reported as before. Note that only aggregated values are available for the approach of Gendreau et al. [7]. GRASP × ELS outperforms the other methods on all the classes, especially in classes 3 and 4. Details on the results for each instance can be found in Tables A2–A6.

**Table A1**
Class 1 instances.

| | Gendreau et al. [7] | | Zachariadis et al. [8] | | Fuellerer et al. [9] | | GRASP × ELS | |
|---|---|---|---|---|---|---|---|---|
| | s | t | s | t | s | t | s | t |
| **0101** | 278.73 | 2.0 | 278.73 | 2.9 | 278.73 | 0.09 | 278.73 | 0.0 |
| **0201** | 334.96 | 0.0 | 334.96 | 1.4 | 334.96 | 0.05 | 334.96 | 0.0 |
| **0301** | 359.77 | 3.5 | 358.40 | 3.8 | 358.4 | 0.23 | 358.4 | 0.0 |
| **0401** | 430.88 | 0.1 | 430.88 | 1.0 | 430.88 | 0.27 | 430.88 | 0.0 |
| **0501** | 375.28 | 1.4 | 375.28 | 1.3 | 375.28 | 0.32 | 375.28 | 0.0 |
| **0601** | 495.85 | 0.3 | 495.85 | 1.9 | 495.85 | 0.30 | 495.85 | 0.0 |
| **0701** | 568.56 | 0.5 | 568.56 | 0.8 | 568.56 | 0.24 | 568.56 | 0.0 |
| **0801** | 568.56 | 0.5 | 568.56 | 0.4 | 568.56 | 0.24 | 568.56 | 0.0 |
| **0901** | 607.65 | 0.4 | 607.65 | 1.2 | 607.65 | 0.57 | 607.65 | 0.0 |
| **1001** | 538.79 | 6.1 | 535.80 | 5.9 | 535.8 | 2.27 | 535.80 | 0.0 |
| **1101** | 505.01 | 2.5 | 505.01 | 3.8 | 505.01 | 0.81 | 505.01 | 0.0 |
| **1201** | 610.57 | 28.5 | 610.00 | 6.3 | 610.00 | 1.54 | 610.00 | 0.2 |
| **1301** | 2006.34 | 29.9 | 2006.34 | 5.8 | 2006.34 | 1.26 | 2006.34 | 0.0 |
| **1401** | 837.67 | 22.2 | 837.67 | 17.1 | 837.67 | 4.10 | 837.67 | 0.2 |
| **1501** | 837.67 | 1.7 | 837.67 | 7.9 | 837.67 | 2.83 | 837.67 | 0.0 |
| **1601** | 698.61 | 2.7 | 698.61 | 13.0 | 698.61 | 1.97 | 698.61 | 0.0 |
| **1701** | 862.62 | 59.0 | 863.27 | 32.9 | 861.79 | 3.28 | 861.79 | 0.0 |
| **1801** | 723.54 | 81.9 | 730.85 | 47.1 | 723.54 | 9.51 | 723.54 | 8.3 |
| **1901** | 524.61 | 128.8 | 524.61 | 100.2 | 524.61 | 7.94 | 524.61 | 0.3 |
| **2001** | 241.97 | 253.6 | 244.54 | 198.3 | 241.97 | 56.06 | 241.97 | 4.5 |
| **2101** | 688.18 | 325.0 | 687.6 | 221.5 | 690.2 | 26.45 | 687.60 | 1.4 |
| **2201** | 740.66 | 2070.7 | 740.66 | 662.9 | 742.91 | 57.43 | 740.66 | 2.1 |
| **2301** | 860.47 | 2210.1 | 839.07 | 1531.4 | 845.34 | 55.94 | 835.26 | 3391.3 |
| **2401** | 1048.91 | 866.9 | 1035.33 | 1012.7 | 1030.25 | 49.77 | 1026.6 | 53.3 |
| **2501** | 830.26 | 2371.0 | 829.45 | 953.8 | 830.82 | 167.14 | 827.39 | 2.4 |
| **2601** | 819.56 | 3597.6 | 819.56 | 1031.7 | 819.56 | 175.69 | 819.56 | 0.4 |
| **2701** | 1099.95 | 355.9 | 1097.63 | 871.2 | 1100.22 | 190.52 | 1082.65 | 486.5 |
| **2801** | 1078.27 | 985.2 | 1042.12 | 781.4 | 1062.23 | 252.48 | 1042.12 | 129.8 |
| **2901** | 1179.01 | 3080.0 | 1188.15 | 1641.9 | 1168.13 | 769.14 | 1162.96 | 549.6 |
| **3001** | 1061.55 | 1834.4 | 1037.05 | 873.3 | 1041.05 | 310.25 | 1033.42 | 2165.9 |
| **3101** | 1464.04 | 288.8 | 1421.2 | 631.4 | 1341.89 | 521.84 | 1306.07 | 5096.1 |
| **3201** | 1352.61 | 1780.8 | 1328.68 | 905.5 | 1334.26 | 517.68 | 1303.52 | 4492.4 |
| **3301** | 1361.51 | 2531.7 | 1328.19 | 1708.6 | 1331.69 | 476.63 | 1301.06 | 4842.1 |
| **3401** | 858.94 | 1941.9 | 719.91 | 834.1 | 712.32 | 614.53 | 713.51 | 3007.4 |
| **3501** | 992.86 | 766.7 | 877.04 | 907.2 | 868.12 | 1452.58 | 870.63 | 2616.5 |
| **3601** | 678.87 | 1530.9 | 594.10 | 1492.6 | 616.69 | 1588.25 | 592.87 | 5264.7 |

**Table A2**
Class 2 instances.

| | Gendreau et al. [7] | | Zachariadis et al. [8] | | Fuellerer et al. [9] | | GRASP × ELS | |
|---|---|---|---|---|---|---|---|---|
| | s | t | s | t | s | t | s | t |
| **0102** | / | / | 305.92 | / | 284.52 | 1.18 | 284.42 | 0.2 |
| **0202** | / | / | 334.96 | / | 334.96 | 0.14 | 334.96 | 0.0 |
| **0302** | / | / | 401.81 | / | 387.70 | 1.29 | 387.70 | 0.8 |
| **0402** | / | / | 440.94 | / | 430.88 | 0.98 | 430.88 | 0.3 |
| **0502** | / | / | 381.85 | / | 375.28 | 7.27 | 375.28 | 0.1 |
| **0602** | / | / | 498.16 | / | 495.85 | 1.77 | 495.85 | 0.4 |
| **0702** | / | / | 741.91 | / | 725.46 | 3.99 | 725.46 | 0.3 |
| **0802** | / | / | 718.18 | / | 709.39 | 8.45 | 674.55 | 0.2 |
| **0902** | / | / | 607.65 | / | 607.65 | 2.39 | 607.65 | 0.2 |
| **1002** | / | / | 708.63 | / | 689.68 | 30.29 | 689.68 | 6.1 |
| **1102** | / | / | 719.56 | / | 711.08 | 22.63 | 693.45 | 25.9 |
| **1202** | / | / | 628.86 | / | 610.57 | 4.26 | 610.57 | 5.4 |
| **1302** | / | / | 2705.05 | / | 2588.81 | 39.22 | 2585.72 | 105.3 |
| **1402** | / | / | 1117.24 | / | 1038.68 | 92.42 | 1038.09 | 177.5 |
| **1502** | / | / | 1099.75 | / | 1021.00 | 73.43 | 1013.29 | 482.8 |
| **1602** | / | / | 702.70 | / | 698.61 | 6.30 | 698.61 | 0.9 |
| **1702** | / | / | 870.86 | / | 870.86 | 4.68 | 870.86 | 53.1 |
| **1802** | / | / | 1065.3 | / | 1030.64 | 176.61 | 1004.99 | 885.9 |
| **1902** | / | / | 796.87 | / | 767.41 | 58.94 | 754.53 | 440.6 |
| **2002** | / | / | 569.20 | / | 534.95 | 726.15 | 537.88 | 2904.5 |
| **2102** | / | / | 1076.24 | / | 1013.49 | 589.8 | 992.83 | 942.9 |
| **2202** | / | / | 1088.33 | / | 1052.85 | 400.88 | 1036.11 | 1741.9 |
| **2302** | / | / | 1124.60 | / | 1043.99 | 1191.5 | 1041.04 | 1226.9 |
| **2402** | / | / | 1234.03 | / | 1188.09 | 238.22 | 1190.70 | 515.6 |
| **2502** | / | / | 1500.07 | / | 1430.31 | 834.92 | 1419.42 | 3154.8 |

Table A2. (*continued*)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **2602** | / | / | 1387.3 | / | 1298.02 | 1025.08 | 1285.01 | 2314.6 |
| **2702** | / | / | 1402.42 | / | 1336.67 | 924.97 | 1327.06 | 4162.1 |
| **2802** | / | / | 2856.93 | / | 2650.06 | 3600.00 | 2587.23 | 4473.9 |
| **2902** | / | / | 2362.75 | / | 2260.47 | 3600.00 | 2212.22 | 3025.5 |
| **3002** | / | / | 1929.93 | / | 1840.56 | 3600.00 | 1816.05 | 4969.2 |
| **3102** | / | / | 2456.28 | / | 2325.98 | 3600.00 | 2311.11 | 5207.1 |
| **3202** | / | / | 2465.17 | / | 2319.31 | 3600.00 | 2322.17 | 5083.2 |
| **3302** | / | / | 2508.68 | / | 2326.13 | 3600.00 | 2285.94 | 5000.4 |
| **3402** | / | / | 1268.93 | / | 1220.53 | 3600.00 | 1212.04 | 5020.6 |
| **3502** | / | / | 1464.93 | / | 1416.88 | 3600.00 | 1419.37 | 5315.5 |
| **3602** | / | / | 1854.06 | / | 1787.01 | 3600.00 | 1782.99 | 4608.7 |

**Table A3**
Class 3 instances.

| | **Gendreau et al. [7]** | | **Zachariadis et al. [8]** | | **Fuellerer et al. [9]** | | **GRASP × ELS** | |
|---|---|---|---|---|---|---|---|---|
| | **s** | **t** | **s** | **t** | **s** | **t** | **s** | **t** |
| **0103** | / | / | 299.70 | / | 296.87 | 8.93 | 284.52 | 0.9 |
| **0203** | / | / | 355.65 | / | 352.16 | 0.44 | 352.16 | 0.1 |
| **0303** | / | / | 409.17 | / | 394.72 | 2.08 | 394.72 | 0.4 |
| **0403** | / | / | 446.61 | / | 445.49 | 1.37 | 430.88 | 0.3 |
| **0503** | / | / | 387.89 | / | 381.69 | 10.34 | 381.69 | 0.2 |
| **0603** | / | / | 499.08 | / | 499.08 | 3.42 | 498.16 | 0.6 |
| **0703** | / | / | 706.99 | / | 701.08 | 4.10 | 678.75 | 0.2 |
| **0803** | / | / | 749.70 | / | 740.85 | 6.88 | 738.43 | 0.6 |
| **0903** | / | / | 622.16 | / | 607.65 | 2.00 | 607.65 | 0.3 |
| **1003** | / | / | 655.70 | / | 624.62 | 35.89 | 615.68 | 0.8 |
| **1103** | / | / | 746.12 | / | 723.00 | 23.49 | 706.73 | 4.5 |
| **1203** | / | / | 610.00 | / | 610.00 | 2.08 | 610.00 | 54.1 |
| **1303** | / | / | 2542.86 | / | 2470.42 | 33.32 | 2454.37 | 20.2 |
| **1403** | / | / | 1092.10 | / | 1018.75 | 104.56 | 996.25 | 28.1 |
| **1503** | / | / | 1186.61 | / | 1171.35 | 73.10 | 1154.66 | 248.9 |
| **1603** | / | / | 698.61 | / | 698.61 | 5.26 | 698.61 | 2.9 |
| **1703** | / | / | 861.79 | / | 861.79 | 3.41 | 861.79 | 2.3 |
| **1803** | / | / | 1124.54 | / | 1091.89 | 135.52 | 1069.45 | 110.4 |
| **1903** | / | / | 816.77 | / | 786.43 | 53.55 | 771.74 | 155.3 |
| **2003** | / | / | 557.72 | / | 544.12 | 375.47 | 524.81 | 1824.2 |
| **2103** | / | / | 1191.07 | / | 1148.02 | 250.91 | 1121.84 | 759.1 |
| **2203** | / | / | 1110.73 | / | 1075.55 | 305.15 | 1052.98 | 1189.9 |
| **2303** | / | / | 1141.51 | / | 1098.70 | 298.37 | 1081.48 | 1288.2 |
| **2403** | / | / | 1136.1 | / | 1116.98 | 155.45 | 1083.14 | 796.3 |
| **2503** | / | / | 1476.14 | / | 1409.5 | 777.12 | 1374.68 | 2539.3 |
| **2603** | / | / | 1436.55 | / | 1384.75 | 759.12 | 1344.66 | 2170.3 |
| **2703** | / | / | 1476.73 | / | 1398.52 | 560.96 | 1378.01 | 1343.6 |
| **2803** | / | / | 2867.46 | / | 2740.68 | 3600.00 | 2629.38 | 5289.0 |
| **2903** | / | / | 2249.8 | / | 2184.45 | 3600.00 | 2107.87 | 3895.1 |
| **3003** | / | / | 2038.55 | / | 1894.16 | 3600.00 | 1850.78 | 5126.6 |
| **3103** | / | / | 2478.94 | / | 2366.77 | 3600.00 | 2305.51 | 5107.4 |
| **3203** | / | / | 2422.98 | / | 2327.25 | 3600.00 | 2267.82 | 5255.0 |
| **3303** | / | / | 2595.41 | / | 2470.07 | 3600.00 | 2390.58 | 4853.6 |
| **3403** | / | / | 1298.48 | / | 1259.88 | 3600.00 | 1237.27 | 5168.7 |
| **3503** | / | / | 1570.67 | / | 1511.42 | 3600.00 | 1477.05 | 5165.8 |
| **3603** | / | / | 1965.46 | / | 1891.90 | 3600.00 | 1834.97 | 5069.6 |

**Table A4**
Class 4 instances.

| | **Gendreau et al. [7]** | | **Zachariadis et al. [8]** | | **Fuellerer et al. [9]** | | **GRASP × ELS** | |
|---|---|---|---|---|---|---|---|---|
| | **s** | **t** | **s** | **t** | **s** | **t** | **s** | **t** |
| **0104** | / | / | 296.75 | | 282.95 | 0.96 | 282.95 | 0.0 |
| **0204** | / | / | 342.00 | | 342.00 | 0.13 | 334.96 | 0.1 |
| **0304** | / | / | 368.56 | | 364.45 | 0.96 | 364.45 | 0.2 |
| **0404** | / | / | 447.37 | | 447.37 | 2.64 | 447.37 | 0.1 |
| **0504** | / | / | 383.87 | | 383.88 | 6.74 | 383.87 | 0.2 |
| **0604** | / | / | 504.78 | | 498.32 | 2.38 | 498.32 | 0.5 |
| **0704** | / | / | 703.85 | | 702.45 | 4.79 | 702.45 | 2.0 |
| **0804** | / | / | 711.07 | | 692.47 | 6.01 | 692.47 | 1.6 |

Table A4. (*continued*)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **0904** | / | / | 625.13 | 625.13 | 3.02 | 625.1 | 1.7 |
| **1004** | / | / | 792.30 | 724.77 | 27.76 | 711.01 | 17.8 |
| **1104** | / | / | 843.52 | 816.45 | 24.75 | 786.85 | 10.7 |
| **1204** | / | / | 618.23 | 614.24 | 5.43 | 614.23 | 1.6 |
| **1304** | / | / | 2714.69 | 2607.66 | 41.65 | 2587.63 | 15.5 |
| **1404** | / | / | 994.66 | 985.01 | 84.86 | 981.90 | 5.5 |
| **1504** | / | / | 1258.49 | 1246.54 | 72.61 | 1234.14 | 55.7 |
| **1604** | / | / | 709.27 | 703.35 | 10.22 | 703.35 | 12.00 |
| **1704** | / | / | 861.79 | 861.79 | 4.08 | 861.79 | 29.7 |
| **1804** | / | / | 1171.51 | 1124.37 | 138.46 | 1118.71 | 235 |
| **1904** | / | / | 819.79 | 798.33 | 58.19 | 778.35 | 350.2 |
| **2004** | / | / | 576.92 | 553.03 | 271.42 | 547.95 | 720.4 |
| **2104** | / | / | 1019.74 | 1001.14 | 365.03 | 978.82 | 1544.9 |
| **2204** | / | / | 1119.34 | 1093.16 | 221.98 | 1045.91 | 673.5 |
| **2304** | / | / | 1123.17 | 1089.66 | 281.65 | 1080.02 | 1523.1 |
| **2404** | / | / | 1160.92 | 1133.98 | 174.27 | 1111.27 | 178.4 |
| **2504** | / | / | 1486.54 | 1441.11 | 669.36 | 1405.65 | 2246.1 |
| **2604** | / | / | 1491.00 | 1451.71 | 1490.06 | 1405.57 | 2913.8 |
| **2704** | / | / | 1397.75 | 1362.87 | 585.99 | 1326.16 | 2643.8 |
| **2804** | / | / | 2770.05 | 2716.94 | 3600.00 | 2654.75 | 5258 |
| **2904** | / | / | 2427.95 | 2350.62 | 3600.00 | 2270.44 | 4406.9 |
| **3004** | / | / | 1965.45 | 1902.68 | 3600.00 | 1856.54 | 3936.3 |
| **3104** | / | / | 2585.67 | 2495.39 | 3600.00 | 2436.42 | 4538.8 |
| **3204** | / | / | 2432.49 | 2362.22 | 3600.00 | 2308.4 | 3908.2 |
| **3304** | / | / | 2601.34 | 2504.63 | 3600.00 | 2416.77 | 5388 |
| **3404** | / | / | 1279.65 | 1251.87 | 3600.00 | 1235.58 | 5402.9 |
| **3504** | / | / | 1634.63 | 1593.25 | 3600.00 | 1538.30 | 5291.0 |
| **3604** | / | / | 1803.86 | 1771.31 | 3600.00 | 1728.69 | 4785.5 |

**Table A5**
Class 5 instances.

| | **Gendreau et al. [7]** | | **Zachariadis et al. [8]** | | **Fuellerer et al. [9]** | | **GRASP × ELS** | |
|---|---|---|---|---|---|---|---|---|
| | **s** | **t** | **s** | **t** | **s** | **t** | **s** | **t** |
| **0105** | / | / | 280.60 | | 278.73 | 0.41 | 278.73 | 2.5 |
| **0205** | / | / | 334.96 | | 334.96 | 0.04 | 334.96 | 0 |
| **0305** | / | / | 358.40 | | 358.40 | 0.40 | 358.40 | 0.4 |
| **0405** | / | / | 430.88 | | 430.88 | 0.74 | 430.88 | 0.2 |
| **0505** | / | / | 375.28 | | 375.28 | 1.36 | 375.28 | 0.1 |
| **0605** | / | / | 495.85 | | 495.85 | 0.56 | 495.85 | 0.0 |
| **0705** | / | / | 661.22 | | 658.64 | 6.41 | 657.77 | 3.0 |
| **0805** | / | / | 643.43 | | 621.85 | 18.79 | 609.9 | 0.7 |
| **0905** | / | / | 607.65 | | 607.65 | 1.10 | 607.65 | 0.2 |
| **1005** | / | / | 695.37 | | 691.04 | 26.46 | 686.78 | 35.9 |
| **1105** | / | / | 652.42 | | 636.77 | 23.67 | 636.77 | 4.2 |
| **1205** | / | / | 610.23 | | 610.23 | 3.27 | 610.23 | 6.4 |
| **1305** | / | / | 2434.99 | | 2416.04 | 42.31 | 2334.78 | 171 |
| **1405** | / | / | 943.08 | | 922.58 | 104.25 | 921.45 | 108.5 |
| **1505** | / | / | 1246.46 | | 1230.22 | 56.51 | 1176.68 | 243.4 |
| **1605** | / | / | 698.61 | | 698.61 | 2.89 | 698.61 | 8.4 |
| **1705** | / | / | 862.62 | | 861.79 | 3.79 | 861.79 | 1.3 |
| **1805** | / | / | 945.88 | | 926.34 | 200.06 | 925.72 | 422.5 |
| **1905** | / | / | 674.20 | | 656.03 | 71.2 | 652.15 | 128 |
| **2005** | / | / | 503.01 | | 480.59 | 420.29 | 480.1 | 1184.2 |
| **2105** | / | / | 914.68 | | 897.55 | 414.62 | 884.84 | 2556.2 |
| **2205** | / | / | 986.02 | | 956.42 | 396.93 | 950.79 | 254.9 |
| **2305** | / | / | 975.42 | | 956.55 | 300.97 | 950.09 | 1456.3 |
| **2405** | / | / | 1065.41 | | 1049.76 | 88.83 | 1046.63 | 430.8 |
| **2505** | / | / | 1212.73 | | 1182.14 | 1175.34 | 1180.57 | 3930.6 |
| **2605** | / | / | 1267.68 | | 1250.41 | 859.96 | 1234.39 | 1798.1 |
| **2705** | / | / | 1309.5 | | 1271.08 | 780.39 | 1262.93 | 2717 |
| **2805** | / | / | 2453.59 | | 2412.8 | 3600.00 | 2368.88 | 5241 |
| **2905** | / | / | 2220.32 | | 2191.56 | 3600.00 | 2175.31 | 5187.1 |
| **3005** | / | / | 1625.42 | | 1570.75 | 3600.00 | 1578.41 | 4982.5 |
| **3105** | / | / | 2132.92 | | 2080.25 | 3600.00 | 2076.07 | 5099.3 |
| **3205** | / | / | 2086.13 | | 2039.14 | 3600.00 | 2034.68 | 5356 |
| **3305** | / | / | 2117.72 | | 2050.72 | 3600.00 | 2046.00 | 4713.7 |
| **3405** | / | / | 1086.79 | | 1070.28 | 3600.00 | 1079.61 | 5385.9 |
| **3505** | / | / | 1324.89 | | 1301.27 | 3600.00 | 1306.19 | 4289.7 |
| **3605** | / | / | 1582.25 | | 1570.81 | 3600.00 | 1572.49 | 5032.1 |

**Table A6**
Class 2–5 instance.

| | Gendreau et al. [7] | | Zachariadis et al. [8] | | Fuellerer et al. [9] | | GRASP × ELS | |
|---|---|---|---|---|---|---|---|---|
| | s | t | s | t | s | t | s | t |
| **01** | 291.60 | 4.2 | 295.74 | 2.2 | 285.77 | 2.87 | 282.65 | 0.9 |
| **02** | 341.02 | 0.1 | 341.89 | 1.3 | 341.02 | 0.1875 | 339.26 | 0.1 |
| **03** | 377.35 | 1.6 | 384.49 | 0.7 | 376.32 | 1.1825 | 376.32 | 0.5 |
| **04** | 437.45 | 0.5 | 441.45 | 2.2 | 438.65 | 1.4325 | 435.01 | 0.2 |
| **05** | 380.20 | 5.0 | 382.22 | 4.7 | 379.03 | 6.4275 | 379.03 | 0.1 |
| **06** | 501.02 | 7.2 | 499.47 | 4.4 | 497.27 | 2.0325 | 497.04 | 0.4 |
| **07** | 700.34 | 6.3 | 703.49 | 4.5 | 696.91 | 4.8225 | 691.11 | 1.4 |
| **08** | 694.99 | 11.2 | 705.60 | 6.4 | 691.14 | 10.0325 | 678.84 | 0.8 |
| **09** | 619.69 | 3.6 | 615.65 | 5.1 | 612.02 | 2.1275 | 612.01 | 0.6 |
| **10** | 700.39 | 36.0 | 713.00 | 9.5 | 682.53 | 30.10 | 675.79 | 15.1 |
| **11** | 739.04 | 55.7 | 740.04 | 18.1 | 721.82 | 23.635 | 705.95 | 11.3 |
| **12** | 620.62 | 49.0 | 616.83 | 61.9 | 611.26 | 3.76 | 611.26 | 16.9 |
| **13** | 2598.2 | 57.5 | 2599.40 | 44.4 | 2520.73 | 39.125 | 2490.62 | 78.0 |
| **14** | 1047.72 | 375.8 | 1036.77 | 167.4 | 991.26 | 96.5225 | 984.42 | 79.9 |
| **15** | 1201.38 | 156.7 | 1197.83 | 86.1 | 1167.28 | 68.9125 | 1144.69 | 257.7 |
| **16** | 702.03 | 20.5 | 702.30 | 78.3 | 699.80 | 6.1675 | 699.79 | 6.0 |
| **17** | 866.37 | 64.9 | 864.26 | 26.4 | 864.06 | 3.99 | 864.05 | 21.6 |
| **18** | 1085.84 | 589.3 | 1076.81 | 250.7 | 1043.31 | 162.6625 | 1029.71 | 413.5 |
| **19** | 772.25 | 633.7 | 776.91 | 376.5 | 752.05 | 60.47 | 739.19 | 268.5 |
| **20** | 564.67 | 954.5 | 551.71 | 518.7 | 528.17 | 448.3325 | 522.68 | 1658.3 |
| **21** | 1066.21 | 460.1 | 1050.43 | 129.0 | 1015.05 | 405.09 | 994.58 | 1450.8 |
| **22** | 1087.46 | 1191.2 | 1076.11 | 941.1 | 1044.49 | 331.235 | 1021.45 | 965.0 |
| **23** | 1104.72 | 2032.4 | 1091.17 | 1000.8 | 1047.23 | 518.1225 | 1038.16 | 1373.6 |
| **24** | 1187.62 | 1454.1 | 1149.12 | 553.5 | 1122.2 | 164.1925 | 1107.93 | 480.3 |
| **25** | 1436.09 | 1205.8 | 1418.87 | 635.9 | 1365.77 | 864.185 | 1345.08 | 2967.7 |
| **26** | 1404.49 | 1173.9 | 1395.63 | 875.3 | 1346.22 | 1033.555 | 1317.41 | 2299.2 |
| **27** | 1450.18 | 521.3 | 1396.60 | 492.5 | 1342.28 | 713.0775 | 1323.54 | 2716.6 |
| **28** | 2738.31 | 2051.2 | 2737.01 | 1079.1 | 2630.12 | 3600.00 | 2560.06 | 5065.5 |
| **29** | 2474.33 | 1406.5 | 2315.20 | 1059.0 | 2246.78 | 3600.00 | 2191.46 | 4128.6 |
| **30** | 1948.72 | 1185.4 | 1889.84 | 1711.2 | 1802.04 | 3600.00 | 1775.44 | 4753.7 |
| **31** | 2506.99 | 2375.8 | 2413.45 | 2500.7 | 2317.10 | 3600.00 | 2282.28 | 4988.2 |
| **32** | 2486.43 | 1664.8 | 2351.69 | 2240.1 | 2261.98 | 3600.00 | 2233.27 | 4900.6 |
| **33** | 2504.00 | 1843.2 | 2455.79 | 2074.1 | 2337.89 | 3600.00 | 2284.82 | 4988.9 |
| **34** | 1466.06 | 1359.1 | 1233.46 | 2549.7 | 1200.64 | 3600.00 | 1191.13 | 5244.5 |
| **35** | 1765.30 | 2061.7 | 1498.78 | 2964.5 | 1455.70 | 3600.00 | 1435.22 | 5015.5 |
| **36** | 1909.88 | 2265.8 | 1801.41 | 2680.3 | 1755.26 | 3600.00 | 1729.79 | 4874 |

### 3.5. *Example of a 2L-CVRP solution*

Let us consider the instance 01 from class 3 (referred as 0103 in Table A3). The solution of value 284.52 corresponds to the best ever published solution with 3 trips (see Fig. 9). Figs. 9(b)–(d) provide a graphical representation of the three packing solutions for the three trips involved in the solution.

## 4. Concluding remarks

This article considers an extension of the well-known CVRP in which two dimensional packing constraints must be addressed in each trip servicing customers. This problem deals with two combinatorial optimization problems: vehicle routing and two-dimensional bin packing. The initial 2L-CVRP is first relaxed into the easier RCPSP-CVRP. The relaxation problem is solved using the GRASP × ELS framework. At the end of GRASP × ELS, the solution is transformed back into a 2L-CVRP solution by packing the items into the vehicles. To our knowledge, this is an innovative approach. The results show that our method is highly efficient and outperforms the best previous published methods on the topic. We are currently investigating the 3L-CVRP, the sequence-dependant 2L-CVRP and the non-orientated cases.

## Acknowledgement

Special thanks to referees who provided us constant support and help in a previous version of this article.

## Appendix

See Tables A1–A6.

## References


[1] Eksioglu B, Vural AV, Reisman A. The vehicle routing problem: a taxonomic review. Computers and Industrial Engineering 2008;57:1472–83.
[2] Current JR, Marsh M. Multiobjective transportation network design and routing problems: taxonomy and annotation. European Journal of Operational Research 1993;65:4–19.
[3] Baldacci R, Battarra M, Vigo D. Routing a heterogeneous fleet of vehicles. In: Golden B, Wasil E, editors. The vehicle routing problem: latest advances and new challenges; 2008. p. 3–27.
[4] Toth P, Vigo D. An overview of vehicle routing problems. In: Toth P, Vigo D, editors. The vehicle routing problem. SIAM monographs on discrete mathematics and applications. Philadelphia, 2002; p. 1–26.
[5] Prins C. Two Memetic algorithms for heterogeneous fleet vehicle routing problems. Engineering Applications of Artificial Intelligence 2009;22:916–28.
[6] Cordeau JF, Gendreau M, Hertz A, Laporte G, Sormany JS. New heuristics for the vehicle routing problem. In: Langevin A, Riopel D, editors. Logistic systems: design and optimization. New York: Wiley; 2005. p. 279–98.
[7] Gendreau M, Iori M, Laporte G, Martello S. A Tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. Networks 2008;51(1): 4–18.
[8] Zachariadis EE, Tarantilis CD, Kiranoudis C. A guided Tabu search for the vehicle routing problem with two dimensional loading constraints. European Journal of Operational Research 2009;3(16):729–43.
[9] Fuellerer G, Doerner KF, Hartl RF, Iori M. Ant colony optimization for the two-dimensional loading vehicle routing problem. Computers and Operations Research 2009;36(3):655–73.
[10] Iori M, Salazar González JJ, Vigo D. An exact approach for capacitated vehicle routing problems with two-dimensional loading constraints. Transportation Science 2007;41(2):253–64.
[11] Hadjiconstantinou E, Christofides N. An exact algorithm for general, orthogonal, two-dimensional knapsack problem. European Journal of Operational Research 1995;83:39–56.
[12] Martello S, Vigo D. Exact solution of the two-dimensional finite bin packing problem. Management Science 1988;44:388–99.
[13] Beldiceanu N, Carlsson M. Sweep as a generic pruning technique applied to the non-overlapping rectangles constraints. Principles and practice of Constraint Programming (CP'2001). Lecture Notes in Computer Science 2001;2239:377–91.
[14] Pisinger D, Sigurd M. On using decomposition techniques and constraint programming for solving the two-dimensional bin packing problem. Technical Report, Department of Computer Science, University of Copenhagen, 2003.
[15] Fekete S, Schepers J, van der Veen J. An exact algorithm for higher-dimensional orthogonal packing. Operations Research 2006;55(3):569–87.
[16] Fekete S, Schepers J. A combinatorial characterization of higher-dimensional orthogonal packing. Mathematics of Operations Research 2004;29:353–68.
[17] Leung TW, Chan CK, Troutt MD. Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem. European Journal of Operational Research 2003;145(3):530–42.
[18] Gonçalves JF. A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. European Journal of Operational Research 2007;183(3): 1212–29.
[19] Hopper E, Turton B. A genetic algorithm for a 2D industrial packing problem. Computers & Industrial Engineering 1999;37(1–2):375–8.
[20] Blazewicz J, Lenstra JK, Rinooy Kan AHK. Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics 1983;5:11–24.
[21] Brucker P, Drexl A, Mohring R, Neumann K, Pesch E. Resource-constrained project scheduling: notation, classification, models, and methods. European Journal of Operational Research 1999;112(1):3–41.
[22] Herroelen W, De Reyck B, Demeulemeester E. Resource-constrained project scheduling: a survey of recent developments. Computers and Operations Research 1998;25(4):279–302.
[23] Kolisch R, Padman R. An integrated survey of deterministic project scheduling. Omega 2001;29:249–72.
[24] Weglarz J. Project scheduling. Recent models, algorithms and applications. Kluwer Academic Publishers; 1999.
[25] Demeulemeester E, Herroelen W. Project scheduling: a research handbook. Kluwer's International Series, International series in operations research & management science, vol. 49. Kluwer Academic Publishers; 1-40207-051-9.
[26] Artigues C, Michelon P, Reusser S. Insertion techniques for static and dynamic resource-constrained project scheduling. European Journal of Operational Research 2003;149:249–67.
[27] Tseng LY, Chen SC. A hybrid metaheuristic for the resource-constrained project scheduling problem. European Journal of Operational Research 2006;175:707–21.
[28] Hartmann S. Packing problems and project scheduling models: an integrating perspective. Journal of the Operational Research Society 2000;51:1083–92.
[29] Toussaint H. Algorithmique rapide pour les problèmes de tournées et d'ordonnancement. Thèse de l'Université Blaise Pascal, Ecole Doctorale Sciences pour l'Ingénieur, D.U. 2053, 2010.
[30] Prins C. A GRASP $\times$ evolutionary local search hybrid for the vehicle routing problem. In: Pereira FB, Tavares J, editors. Bio-inspired algorithms for the vehicle routing problem, Studies in computational intelligence, vol. 161. Berlin: Springer; 2009. p. 35–53.
[31] Feo TA, Resende MGC. Greedy randomized adaptive search procedures. Journal of Global Optimization 1995;6:109–33.
[32] Lacomme P, Prins C, Ramdane-Chérif W. Competitive Memetic algorithms for arc routing problems. Annals of Operations Research 2004;131:159–85.
[33] Prins C. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers and Operations Research 2004;31:1985–2002.
[34] Duhamel C, Lacomme P, Prins C, Prodhon C. A GRASPxELS approach for the capacitated location-routing problem. Computers and Operations Research 2010;37(11):1912–23.
[35] Duhamel C, Lacomme P, Prins C, Prodhon C. A memetic approach for the capacitated location routing problem. EU-MEeting 2008, Troyes, France, October 23–24, 2008.
[36] Clarke G, Wright JW. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 1964;12:568–81.
[37] Golden BL, DeArmon JS, Baker EK. Computational experiments with algorithms for a class of routing problems. Computers and Operations Research 1983;10(1):47–59.
[38] Kolisch R. Experimental investigation of heuristics for resource-constrained project scheduling: an update. European Journal of Operational Research 2006;174:23–37.
[39] Beasley JE. Route-first cluster-second methods for vehicle routing. Omega 1983;11:403–8.
[40] Martello S, Pisinger D, Vigo D. The three-dimensional bin packing problem. Operations Research 2000;48:256–67.
[41] Martello S, Pisinger D, Vigo D, Den Boef D, Korst J. Algorithm 864: general and robot-packable variants of the three-dimensional bin packing problem. ACM Transactions on Mathematical Software 2007;33(1). article number 7.