A Tabu Search Algorithm for a Routing and Container Loading Problem

4 authors, including:

Michel Gendreau
Polytechnique Montréal
552 PUBLICATIONS   **40,957** CITATIONS

SEE PROFILE

Manuel Iori
Università degli Studi di Modena e Reggio Emilia
162 PUBLICATIONS   **5,402** CITATIONS

SEE PROFILE

A Tabu Search Algorithm for a Routing and Container Loading Problem

Michel Gendreau
Centre de recherche sur les transports, Université de Montréal, C.P. 6128 succursale Centre-ville,
Montréal, H3C 3J7 Canada, michelg@crt.umontreal.ca

Manuel Iori
Dipartimento di Elettronica, Informatica e Sistemistica, Università degli Studi di Bologna, Viale Risorgimento 2,
40136 Bologna, Italy, miori@deis.unibo.it

Gilbert Laporte
Centre de recherche sur les transports, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine,
Montréal, H3T 2A7 Canada, gilbert@crt.umontreal.ca

Silvano Martello
Dipartimento di Elettronica, Informatica e Sistemistica, Università degli Studi di Bologna, Viale Risorgimento 2,
40136 Bologna, Italy, smartello@deis.unibo.it

This article considers a combination of capacitated vehicle routing and three-dimensional loading, with additional constraints frequently encountered in freight transportation. It proposes a tabu search algorithm that iteratively invokes an inner tabu search procedure for the solution of the loading subproblem. The algorithm is experimentally evaluated both on instances adapted from vehicle routing instances from the literature and on new real-world instances.

1. Introduction

The problem considered in this paper is a combination of vehicle routing and three-dimensional loading, with additional constraints frequently encountered in freight transportation. A shipper must define the routes of its vehicle fleet to send goods, each consisting of a rectangular box of given size and weight, to a number of clients, while minimizing the total travel cost. All goods required by a client must be placed on the same vehicle. All vehicles have the same three-dimensional loading space and the same weight capacity. The solution of the problem calls for the determination, for each vehicle, of a set of goods whose total weight does not exceed the vehicle weight capacity and of a feasible packing of the goods in the loading space. In other words, each vehicle requires the solution of a three-dimensional packing problem, which consists of finding a nonoverlapping packing of a set of rectangular boxes into a rectangular container. We assume that the boxes can be rotated by 90° degrees on the horizontal plane, while upside-down rotations are not allowed. Figure 1 depicts a solution to a simple problem with two vehicles and five clients.

In addition to the above weight and packing constraints, other operational constraints are often

encountered in real-world applications. If some of the goods are fragile, it may be requested that no nonfragile item be placed on top of a fragile one. In addition, when boxes are stacked, one must ensure that the supporting surface is large enough to guarantee the stability of the loading. Finally, it is frequently requested that the loading of each vehicle allow an easy unloading; that is, it should be possible to unload the goods requested by one client without shifting the goods requested by other clients. An example of three-dimensional loading for vehicle 1 of Figure 1 is shown in Figure 2, where it is assumed that the vehicle is unloaded in the direction of the z axis.

The *capacitated vehicle routing problem* (CVRP) is among the most widely investigated problems in combinatorial optimization. We refer the reader to a recent volume (Toth and Vigo 2002) for a thorough review of this research area and, for recent surveys on metaheuristics, to Cordeau et al. (2005) and Cordeau and Laporte (2004). Among very recent papers on the CVRP we mention an ant colony optimization heuristic (Reimann, Doerner, and Hartl 2004) and an exact algorithm obtained by combining branch-and-cut and branch-and-price (Fukasawa et al. 2004).
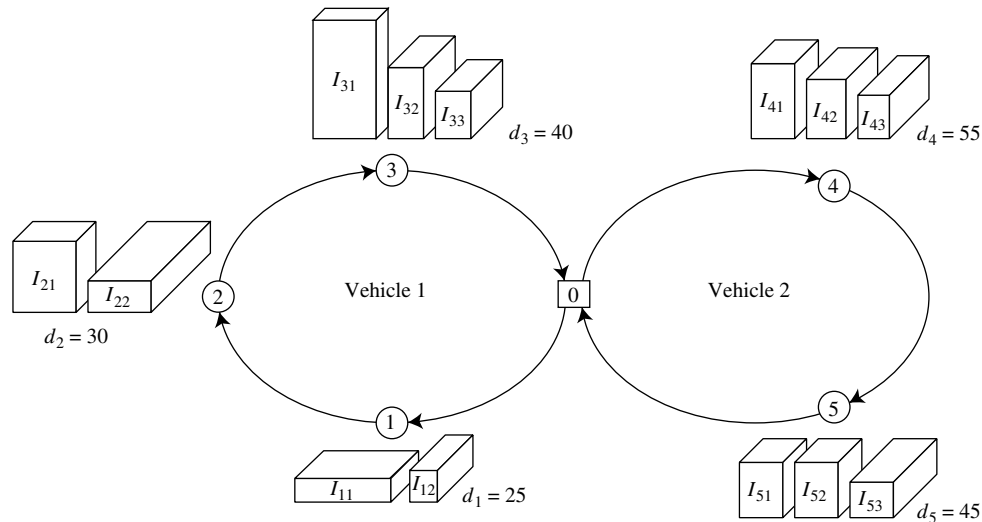
Figure 1   Example of the 3L-CVRP

The loading problem is related to various three-dimensional packing problems, in particular to container-loading problems, where constraints on the supporting surface or on the fragility of the items are frequently considered (e.g., Bortfeldt and Gehring 2001; Eley 2002; Pisinger 2002). Constraints on the loading or unloading sequence are common in the pickup-and-delivery literature (e.g., Levitin and Abezgaouz 2003; Xu et al. 2003).

To our knowledge, no study has so far considered the combination of routing and three-dimensional

Figure 2   Possible Three-Dimensional Loading for Vehicle 1 of Figure 1

loading introduced above. A combination of vehicle routing and two-dimensional loading (the *two-dimensional loading capacitated vehicle routing problem*, 2L-CVRP) was solved in Iori, Salazar, González, and Vigo (2003) through an exact branch-and-cut algorithm and in Gendreau et al. (2006) by means of a metaheuristic algorithm. Using a similar notation, we denote our problem as the *three-dimensional loading capacitated vehicle routing problem* (3L-CVRP).

The problem of deciding whether a set of three-dimensional boxes can be packed into a rectangular container with no additional constraint is already NP-hard (see, e.g., Martello, Pisinger, and Vigo 2000). The same holds for the CVRP and, of course, for the 3L-CVRP.

In this paper we propose a tabu search algorithm for the 3L-CVRP. In §2 we introduce a formal description of the problem. In §3 we describe heuristic and metaheuristic algorithms for a subproblem to be solved by our main algorithm, namely that of finding a feasible loading for a set of clients assigned to a route. Our main tabu search algorithm is presented in §4 and experimentally evaluated in §5, both on instances adapted from CVRP instances from the literature and on new real-world instances.

2.   Problem Description

Let $V = \{0, 1, \ldots, n\}$ be a set of $n + 1$ vertices corresponding to a depot (vertex 0) and n clients (vertices $1, \ldots, n$), and E a set of edges (i, j) connecting all vertex pairs. Let $G = (V, E)$ be the induced graph and denote by c_{ij} the cost of edge (i, j). Let v be the number of available identical vehicles. Each vehicle has a weight capacity D and a three-dimensional rectangular loading space defined by width W, height H, and length L. Denote by $S = W \cdot H \cdot L$ the available
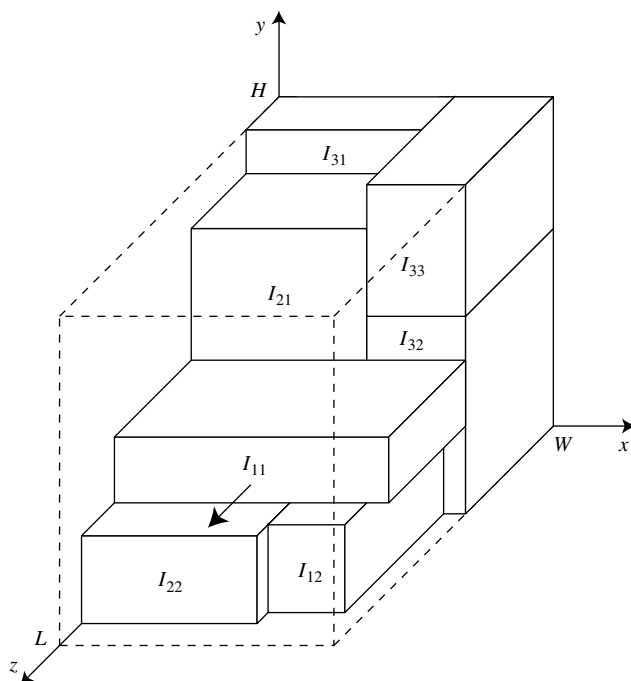
rectangular loading space. Each client i ($i = 1, \ldots, n$) requires a set of m_i three-dimensional items I_{ik} ($k = 1, \ldots, m_i$) having width w_{ik}, height h_{ik}, and length l_{ik}, whose total weight is d_i. Let $s_i = \sum_{k=1}^{m_i} w_{ik} h_{ik} l_{ik}$ denote the total amount of space needed by client i.

The items have a fixed orientation with respect to height, but they can be rotated by 90° on the w-l plane. In addition, each item I_{ik} has a fragility flag f_{ik} ($i = 1, \ldots, n$, $k = 1, \ldots, m_i$), equal to 1 if I_{ik} is fragile and to 0 otherwise: No nonfragile item can be placed over a fragile item, although fragile items can be stacked. When an item I_{ik} is placed over other items, it is necessary to evaluate the corresponding supporting area. Let $\bar{h}$ be the height at which the bottom of I_{ik} is placed and $\bar{A}$ (*supporting area*) the total area of the bottom of I_{ik} that lies on items whose top is at height $\bar{h}$; the placing is feasible only if the supporting area is no less than a given threshold percentage a of the base of the item, that is, if $\bar{A} \geq a w_{ik} l_{ik}$ (note that the constraint is always satisfied for $\bar{h} = 0$). Finally, the loading of each vehicle must obey the following LIFO policy. When client i is visited, it must be possible to unload all items I_{ik} of his demand through a sequence of straight movements (one per item) parallel to the L edge. In other words, no item demanded by a client that is visited later may be placed over I_{ik} or between I_{ik} and the rear of the vehicle.

The 3L-CVRP consists of finding a set of at most v routes (one per vehicle), each one including the depot, such that the following conditions are satisfied:

(1) Each client is served by exactly one vehicle.

(2) No vehicle carries a total weight exceeding its capacity.

(3) For each vehicle there exists a feasible orthogonal three-dimensional loading of all the items demanded by all the served clients, satisfying the constraints on

    (3.a) fixed vertical orientation,
    (3.b) fragility,
    (3.c) supporting area,
    (3.d) LIFO policy.

(4) The overall length of the edges included in the routes is a minimum.

The loading space of a vehicle is represented in the positive orthant of a Cartesian coordinate system, with the W edge, the H edge, and the L edge respectively parallel to the x axis, the y axis, and the z axis. The vehicle rear is the $W \times H$ rectangle at coordinate $(0, 0, L)$ (see Figure 2). The position of an item I_{ik} in a loading pattern is given by the coordinates (x_{ik}, y_{ik}, z_{ik}) of its bottom left back corner; in Figure 2, for example, item I_{31} is packed in the origin of the coordinate system.

We assume that for each client i, the demanded set of items is sorted so that all nonfragile items precede the fragile ones, and that each of these two subsets is sorted by decreasing volume, breaking ties by decreasing height.

3. Heuristics for the Single-Vehicle Loading Problem

Our tabu search algorithm for 3L-CVRP, to be described in the next section, iteratively calls a routine for solving the following subproblem. Given an ordered set of clients $\{i_1, \ldots, i_h\}$ to be visited in this order and the corresponding demand set $\{I_{i_j k}: j = 1, \ldots, h; k = 1, \ldots, m_{i_j}\}$, decide whether all these items can be loaded into a single vehicle while satisfying a set of conditions. In the case of 3L-CVRP, these are conditions (2) and (3) defined in the previous section, but the approach we follow can be easily extended to handle other conditions on the loading.

The problem, denoted hereafter as 3L-SV, is strongly NP-hard because the special case of determining a feasible orthogonal three-dimensional loading of all the items (with no fragile item, zero supporting area, and no need of LIFO policy) is already a strongly NP-hard problem (known as the three-dimensional bin-packing problem; see, e.g., Lodi 2002). We have solved 3L-SV through a tabu search algorithm called $TS_{3L\text{-}SV}$ for the three-dimensional strip-packing problem: Given a container (strip) of width W, height H, and infinite length, the algorithm determines a feasible loading of minimum length, to be tested against L.

Algorithm $TS_{3L\text{-}SV}$ makes use of two greedy heuristics for the neighborhood exploration. If no feasible solution is found, the algorithm returns an infeasible solution, requiring a loading length $\lambda > L$, used by the outer tabu search for evaluating the move. If the total weight or the total volume of the demand set exceeds the vehicle capacities, a dummy loading of length $\lambda = 2L$ is returned.

3.1. Tabu Search

In the $TS_{3L\text{-}SV}$ algorithm, a simple tabu search scheme explores the neighborhood by modifying the sequence in which the items of the demand set are loaded into the vehicle. Given such a sequence, two greedy heuristics, to be described in the next section, are executed to find a feasible loading that minimizes the used length. In other words, each heuristic packs the items in sequence, while satisfying the loading conditions and by using a loading space of maximum width W, maximum height H, and minimum length λ. If $\lambda \leq L$, then we have obtained the required packing and the tabu search terminates. Otherwise, the incumbent solution (corresponding to the minimum $\lambda > L$) is possibly updated and a new sequence is attempted.

Because the inner heuristics pack the items starting from the origin, the first sequence is obtained by listing the clients in the inverse visiting order (recall that,

for each client, the items are initially sorted as described in §2). See, for example, Figure 1, where the underlying sequence is $\{I_{31}, I_{32}, I_{33}, I_{21}, I_{22}, I_{11}, I_{12}\}$ for vehicle 1 and $\{I_{51}, I_{52}, I_{53}, I_{41}, I_{42}, I_{43}\}$ for vehicle 2.

After the first call to the heuristics, if $\lambda > L$, the algorithm classifies the items according to the way in which they have been packed: Those items that have been entirely packed within the vehicle are classified as type 1, while those having a portion outside it (in the direction of z) are classified as type 2. The neighborhood is then explored by considering as possible moves all sequences obtained from the current one by interchanging, each time, a different item pair (i, j), such that i is of type 1 and j is of type 2. For each sequence we execute the heuristics and assign a score to the move. The minimum score move is finally selected. The score is computed as follows. Let $\lambda(i, j)$ be the solution value returned by the heuristics when i and j are interchanged and $\varphi(\xi)$ the ratio of the number of times item ξ has been selected for interchanging in the previous moves to the number of interchanges performed. The score is then $\lambda'(i, j) = \lambda(i, j) + (\varphi(i) + \varphi(j))L$, where the second term penalizes frequent interchanges of the same items, thus increasing diversification. Once the move has been selected and executed, the items are reclassified and a new exploration follows.

The algorithm makes use of two tabu lists, one per item type. A move interchanging i and j is tabu if i is in one list and j in the other one. A tabu move is, however, accepted when it produces a solution improving on the incumbent. A convenient value for the tabu list length was experimentally determined as the minimum between 10 and half the number of items to be loaded. The search terminates when a feasible loading of length $\lambda \leq L$ is found or after a prefixed number of iterations.

3.2. Greedy Heuristics
We have extended to our problem two heuristics developed for two-dimensional packing problems: the bottom left algorithm (see Baker, Coffman, and Rivest 1980) and the touching perimeter algorithm (see Lodi, Martello, and Vigo 1999).

The original bottom left heuristic packs two-dimensional items, one at a time, by placing the current item in the lowest possible position, left justified. Our adaptation, called $\text{BL}_{\text{3L-SV}}$, considers, for the current three-dimensional item, the so-called normal positions (introduced, for the two-dimensional case, in Christofides and Whitlock 1977), that is, those positions in which the bottom (resp. left, resp. back) surface touches either the bottom (resp. left side, resp. back) of the container or the top (resp. right side, resp. front) of an already packed item. The normal positions are scanned according to lowest x values, breaking ties by lowest z and then by lowest y values. For

each position, both feasible orientations on the w-l plane are considered. The first packing, which satisfies loading conditions (3.b)–(3.d) (on fragility, supporting area, and LIFO policy; see §2), is selected. The process is iterated until all items are packed or a loading length exceeding twice the vehicle length has been reached (in which case the algorithm fails and returns $\lambda = 2L$).

The original touching perimeter algorithm packs two-dimensional items, one at a time, in a normal position, selected as the one maximizing the percentage of the item perimeter that touches the container and other items already packed. Our extension, $\text{TA}_{\text{3L-SV}}$ (touching area), evaluates each three-dimensional normal position twice (for the two-item orientations) and, if conditions (3.b)–(3.d) are satisfied, computes the percentage of the item surface touching the container and other items already packed. The position providing the highest percentage is selected. In this case also, we iterate the process until all items are packed or the loading length exceeds twice the vehicle length.

Let $\bar{m}$ be the number of items to be loaded. The number of normal positions is $O(\bar{m})$. For each item and for each position the feasibility check (and, for $\text{TA}_{\text{3L-SV}}$, the touching area computation) requires $O(\bar{m})$ time. It follows that the time complexity of both algorithms is $O(\bar{m}^3)$. In practice, however, algorithm $\text{TA}_{\text{3L-SV}}$ tends to be more time consuming, because it must evaluate all positions before taking a decision, while $\text{BL}_{\text{3L-SV}}$ accepts the first feasible position.

4. A Tabu Search Algorithm for the 3L-CVRP
The tabu search algorithm we have developed for 3L-CVRP makes use of some successful tools introduced for vehicle routing problems in Gendreau, Hertz, and Laporte (1994), and for 2L-CVRP in Gendreau et al. (2006). The algorithm can accept moves producing infeasible solutions, in which some vehicles carry a total weight exceeding capacity D or are assigned a loading exceeding length L. All other requirements must, however, be satisfied.

An initial solution is obtained through one of two algorithms, one for general graphs (HGEN, always executed) and one for Euclidean graphs (HEUCL, only executed in such special cases). These two algorithms were obtained by adapting to our case, respectively, the savings algorithm for the CVRP (see Clarke and Wright 1964) and the algorithm for periodic and multidepot vehicle routing problems presented in Cordeau, Gendreau, and Laporte (1997). In algorithm HGEN, which starts with one client per route and proceeds by merging couples of routes, we normally accept a merging only if the resulting route is

totally feasible, that is, if (i) condition (2) of §2 on the weight capacity is satisfied and (ii) by applying algorithm TS$_{3L-SV}$ of §3.1 we obtain a feasible loading that uses a length $\lambda \leq L$. If, however, the algorithm becomes stuck without obtaining a number of routes equal to the number of vehicles, then in the subsequent iterations we accept mergings in which (i) or (ii) above is not satisfied. Similarly, in algorithm HEUCL, which is based on progressive assignments of clients to routes, we normally accept an assignment only if conditions (i) and (ii) hold, except for the last route, where infeasible assignments are accepted. When both algorithms are executed, the solution with the smaller number of infeasible vehicles is selected.

Starting with the initial solution, we explore the neighborhood by moving a client from one route to another. Both routes in the move are reoptimized through the 4-opt *generalized insertion procedure* (GENI), presented in Gendreau, Hertz, and Laporte (1992) for the travelling salesman problem. Each move assigning client i to vehicle j is evaluated by a score defined, for given parameters α, β, and γ, as

$$score = (\text{route length}) + \alpha(\text{excess weight})$$
$$+ \beta(\text{excess length}) + \gamma f(i, j), \qquad (1)$$

where $f(i, j)$ denotes the ratio between the number of times a move assigned client i to vehicle j and the number of accepted moves. The sum, over the clients assigned to the two routes under consideration, of the weight excesses is directly computed, while that of the length excesses is obtained by executing algorithm TS$_{3L-SV}$ of §3.2. The second and third term in Equation (1) penalize infeasibilities, while the fourth term increases diversification.

Given a client i, assume that the corresponding incident costs c_{ij} ($j = 1, \ldots, n$) are sorted by nondecreasing values; we define as candidates for i the p clients corresponding to the p first costs (p is a prefixed parameter). At each iteration, we consider in turn each client and consider its possible assignment to each route containing at least one of its p candidates. For each of these possible moves we compute the score just defined and select the one yielding the minimum score. When a move is performed, reinserting a client into its former route is declared tabu for ϑ iterations, unless this leads to an improvement of the incumbent.

Good values for the parameters were obtained through computational experiments. For three of them, the best choice turned out to be a fixed value: $p = \min\{n/4, 20\}$, $\vartheta = \min\{n/10, 15\}$, and $\gamma = \sqrt{2nv}$. For α and β, a self-adjusting setting proved to be more effective. We start with $\alpha = 20\bar{c}/D$ and $\beta = 20\bar{c}/L$, where $\bar{c}$ denotes the average edge cost. Whenever a move is performed, let Δ_α and Δ_β denote, respectively,

the variation of the total weight and length excess. If Δ_α (resp. Δ_β) is positive, then we increase α (resp. β) by 10%, while if it is negative, we decrease α (resp. β) by 10%.

The only diversification tool we use is the fourth term in the score computation, while we use two kinds of intensification. Assume the current solution has an edge cost lower than that of the incumbent and there is no weight excess in any route. Then if some routes have a length excess, their loadings are recomputed through algorithm TS$_{3L-SV}$ of §3.2 by doubling the number of allowed iterations. In addition, whenever the incumbent solution is improved, the value of parameter p is doubled for the next iteration.

5. Computational Experiments

The tabu search algorithm described in §§3 and 4 was coded in C and tested on a test set obtained by modifying instances from the literature and on real-world instances. The instances from the literature can be downloaded at www.or.deis.unibo.it/research.html. The experiments were executed on a Pentium IV, 3 GHz with 512 MB of RAM.

5.1. Experiments on Instances Derived from the Literature

In Tables 1 and 2 we present the results obtained by testing our algorithm on instances derived from the literature. In particular, the graphs, the weights demanded by the customers, and the vehicle weight capacities were taken from 27 Euclidean CVRP instances (see Toth and Vigo 2002). The loading space was defined as $W = 25$, $H = 30$, and $L = 60$. For each client i ($i = 1, \ldots, n$), the number m_i of requested items was uniformly and randomly generated between 1 and 3. For each item I_{ik} ($k = 1, \ldots, m_i$), the width w_{ik} (resp. height h_{ik}, resp. length l_{ik}) was uniformly randomly generated in the interval $[0.2W, 0.6W]$ (resp. $[0.2H, 0.6H]$, resp. $[0.2L, 0.6L]$) so as to have both relatively small and relatively large items. For each instance, the number of vehicles was increased, with respect to the value in the original CVRP instance, to ensure that a feasible solution exists. This was obtained by modifying the greedy heuristics of §3.2: Instead of computing a possibly unfeasible single-vehicle solution, the modified algorithms compute a feasible solution that loads the items of all clients into the minimum number of vehicles. The input threshold a for the minimum supporting area (see §2) was set equal to 0.75.

The algorithm was tested in two different versions: single start and multistart. Both versions were assigned a time limit depending on the number of customers: 1,800 CPU seconds for $n \leq 25$ (first group of nine instances), 3,600 CPU seconds for $25 < n < 50$

Table 1   Instances from the Literature

					Single start		Multistart		
Instance	n	M	v	z_0	z	sec_z	z	% *gap*	sec_z
E016-03m	15	32	5	—	316.32	129.5	316.32	0.00	159.5
E016-05m	15	26	5	—	350.58	5.3	350.58	0.00	12.2
E021-04m	20	37	5	—	447.73	461.1	447.73	0.00	1,499.1
E021-06m	20	36	6	—	448.48	181.1	448.48	0.00	1,274.6
E022-04g	21	45	7	650.21	464.24	75.8	464.24	0.00	78.9
E022-06m	21	40	6	604.09	504.46	1,167.9	504.46	0.00	42.1
E023-03g	22	46	6	1,212.57	831.66	181.1	831.66	0.00	292.9
E023-05s	22	43	8	—	871.77	156.1	873.14	0.16	632.3
E026-08m	25	50	8	—	666.10	1,468.5	676.60	1.58	34.7
E030-03g	29	62	10	1,374.45	911.16	714.0	893.61	−1.93	1,130.4
E030-04s	29	58	9	1,252.43	819.36	396.4	818.65	−0.09	778.9
E031-09h	30	63	9	—	651.58	268.1	717.74	10.15	2,120.8
E033-03n	32	61	9	3,990.25	2,928.34	1,639.1	2,816.93	−3.80	2,514.1
E033-04g	32	72	11	—	1,559.64	3,451.6	1,548.41	−0.72	2,973.2
E033-05s	32	68	10	1,781.98	1,452.34	2,327.4	1,488.79	2.51	804.7
E036-11h	35	63	11	—	707.85	2,550.3	714.37	0.92	1,180.6
E041-14h	40	79	14	—	920.87	2,142.5	909.99	−1.18	1,237.9
E045-04f	44	94	14	2,141.86	1,400.52	1,452.9	1,452.02	3.68	2,278.8
E051-05e	50	99	13	1,129.70	871.29	1,822.3	827.99	−4.97	2,174.4
E072-04f	71	147	20	968.56	732.12	790.0	732.12	0.00	795.3
E076-07s	75	155	18	1,668.08	1,275.20	2,370.3	1,226.20	−3.84	1,616.1
E076-08s	75	146	19	1,905.56	1,277.94	1,611.3	1,291.53	1.06	6,708.9
E076-10e	75	150	18	1,715.58	1,258.16	6,725.6	1,271.40	1.05	1,002.9
E076-14s	75	143	18	—	1,307.86	6,619.3	1,317.86	0.82	798.7
E101-08e	100	193	24	2,209.84	1,570.72	5,630.9	1,616.39	2.91	1,727.3
E101-10c	100	199	28	2,943.95	1,847.95	4,123.7	1,839.12	−0.48	1,239.7
E101-14s	100	198	25	—	1,747.52	7,127.2	1,773.50	1.49	1,563.0
Average					1,042.26	2,058.9	1,043.33	0.35	1,358.2

(second group of nine instances), and 7,200 CPU seconds for $n \geq 50$ (last group of nine instances). These values were systematically reached by the algorithm.

The multistart approach operates with the same parameters setting as the single start, but it is assigned a maximum number of iterations, after which the process is halted and reexecuted from scratch. It turns out that, due to the presence of a time limit, the number of iterations performed by the tabu search decreases when the size of the instance increases, as each iteration takes a larger CPU time. Hence the multistart approach was assigned, at each start, a maximum number of iterations depending on the number of customers: 25,000 for $n \leq 25$, 5,000 for $25 < n < 50$, and 1,000 for $n \geq 50$. For a given vehicle, the excess of length is computed by invoking algorithm $TS_{3L\text{-}SV}$ of §3 with a limit of three iterations.

The first five entries in each row of Table 1 give the name of the original CVRP instance (note that the original number of vehicles is in the instance name, after the hyphen), the number of clients (n), the total number of items ($M = \sum_{i=1}^{n} m_i$), the number of vehicles (v), and the initial solution value (z_0) found by the greedy heuristics of §3.2. A dash in the z_0 field indicates that no feasible solution was found by the greedy heuristics. The next two entries

give, for the single-start execution, the final solution value, z, and the elapsed CPU time in seconds, sec_z, when the best solution was found. For the multistart execution (next three entries) we give, in addition, the percentage difference, % *gap*, between the two approaches, computed as $100(z(\text{multistart}) - z(\text{single start}))/z(\text{single start})$.

The two versions have similar performances: Out of 27 instances, in 8 cases identical solutions are identified, in 11 cases single-start outperforms multistart, and the opposite holds in the remaining 8 cases. The single-start approach produces an average solution value (see the last line of the table) that is lower by 0.35%. On the other hand, the time required to find the best solution is higher for the single-start version. Both versions always improve the initial solution found by the heuristics. In particular, for the instances for which the initial solution is feasible, the average improvement is higher than 40%. The next computational experiments were thus carried on for the single-start version alone.

In Table 2 we examine, for the single-start version with the same time limits as in Table 1, the effect of constraints (3.b)–(3.d) of §2 (on fragility, supporting area, and LIFO policy). The first two entries in each row refer to the instances with all conditions imposed;

Table 2   **Solution Values for Different Constraints Configurations**

Instance	All constraints		No fragility		No LIFO		No support		3D-loading only	
	z	sec_z	z	sec_z	z	sec_z	z	sec_z	z	sec_z
E016-03m	316.32	129.5	316.32	1.8	301.74	40.2	301.74	4.7	297.65	3.4
E016-05m	350.58	5.3	334.96	13.0	334.96	3.7	334.96	0.3	334.96	0.6
E021-04m	447.73	461.1	430.02	1,137.9	392.44	639.1	388.10	33.7	362.27	448.1
E021-06m	448.48	181.1	440.68	62.0	430.88	12.3	430.88	22.4	430.88	11.1
E022-04g	464.24	75.8	462.59	60.9	422.90	461.5	435.93	54.5	395.64	0.5
E022-06m	504.46	1,167.9	498.32	90.6	495.85	204.5	498.16	66.9	495.85	14.7
E023-03g	831.66	181.1	801.03	112.1	732.51	7.1	762.63	26.8	742.23	1.8
E023-05s	871.77	156.1	864.54	686.0	810.65	180.9	799.38	71.7	735.14	104.9
E026-08m	666.10	1,468.5	677.06	8.5	630.13	1,518.7	640.94	764.3	630.13	977.8
E030-03g	911.16	714.0	843.33	3,109.3	827.11	1,541.4	803.18	1,881.6	717.90	410.7
E030-04s	819.36	396.4	819.36	1,135.4	767.22	3,440.3	772.55	2,781.2	718.24	208.1
E031-09h	651.58	268.1	669.16	3,042.4	635.15	3,306.0	616.95	1,799.4	614.60	1,302.7
E033-03n	2,928.34	1,639.1	2,753.91	3,371.4	2,549.68	1,308.9	2,591.84	104.1	2,316.56	2,317.3
E033-04g	1,559.64	3,451.6	1,518.26	2,599.5	1,389.31	2,621.9	1,348.19	1,972.6	1,276.60	2,121.3
E033-05s	1,452.34	2,327.4	1,414.19	574.8	1,346.44	2,489.3	1,352.20	2,830.9	1,196.55	2,916.4
E036-11h	707.85	2,550.3	711.11	3,080.6	703.57	2,673.6	704.80	1,339.4	698.61	863.0
E041-14h	920.87	2,142.5	920.87	2,138.3	906.42	697.5	920.87	2,129.1	906.42	753.2
E045-04f	1,400.52	1,452.9	1,433.51	3,100.9	1,331.71	2,000.8	1,245.57	1,848.4	1,124.33	2,198.9
E051-05e	871.29	1,822.3	853.05	3,768.5	781.77	67.8	734.77	6,772.3	680.29	1,390.3
E072-04f	732.12	790.0	653.47	2,364.9	629.77	1,355.3	610.63	779.9	529.00	7,007.5
E076-07s	1,275.20	2,370.3	1,185.67	6,244.3	1,210.78	2,771.5	1,188.60	4,715.0	1,004.40	6,262.5
E076-08s	1,277.94	1,611.3	1,243.22	3,562.1	1,160.67	5,004.2	1,172.20	4,950.0	1,068.96	2,078.7
E076-10e	1,258.16	6,725.6	1,248.25	648.4	1,153.60	1,003.1	1,106.43	2,446.2	1,012.51	4,314.1
E076-14s	1,307.09	6,619.3	1,187.68	3,896.4	1,154.51	1,998.3	1,113.80	5,720.5	1,063.61	1,052.5
E101-08e	1,570.72	5,630.9	1,673.08	7,187.8	1,420.51	3,146.1	1,375.99	2,159.2	1,371.32	500.9
E101-10c	1,847.95	4,123.7	1,775.52	6,887.7	1,605.54	516.6	1,579.47	2,351.4	1,557.12	1,075.0
E101-14s	1,747.52	7,127.2	1,662.10	6,205.3	1,556.33	7,154.0	1,536.49	3,201.9	1,378.52	3,983.2
Average	1,042.26	2,058.9	1,014.49	2,410.8	951.19	1,709.8	939.53	1,882.5	876.31	1,567.4
Av. % *gap*			−2.66		−8.74		−9.86		−15.87	

that is, they give the same solution values and CPU times as Table 1. The four next column pairs refer to the same instance without the fragility constraint, without the LIFO constraint, without the supporting area constraint, and with none of these constraints. Removing the fragility constraint improves the average solution value by 2.66%. A stronger effect is obtained by removing the LIFO constraint and the supporting area constraint: The improvements are 8.74% and 9.86%, respectively. The removal of all three constraints yields an overall average solution value improvement of 15.87%. In this last configura-

tion the CPU time required to achieve the best solution is about 75% of that required for the completely constrained configuration. The removal of constraints always produced a solution value improvement, except for eight cases for the fragility constraint and a single case for the supporting area constraint.

5.2. Experiments on Real-World Instances
The real-world instances considered in this section were provided by an Italian company producing bedroom furniture, such as wardrobes, chests of drawers, night tables, and beds. Deliveries in continental

Table 3   **Real-World Instances**

| Instance	n	M	v	z_0	1 hour CPU time		10 hours CPU time		24 hours CPU time	
					z	sec_z	z	sec_z	z	sec_z
F01	44	141	4	7,711	3,723	2,839.4	3,694	32,133.9	3,694	32,133.9
F02	49	152	4	7,167	4,182	1,993.8	4,182	1,993.8	3,941	86,046.8
F03	55	171	4	6,111	3,674	3,478.5	3,650	31,776.5	3,650	31,776.5
F04	57	159	4	7,059	4,686	2,520.5	4,509	5,995.1	4,509	5,995.1
F05	64	181	4	7,408	7,235	2,366.3	6,886	33,917.9	6,241	75,441.1
Average				7,091	4,700	2,639.7	4,584	21,163.4	4,407	46,278.7
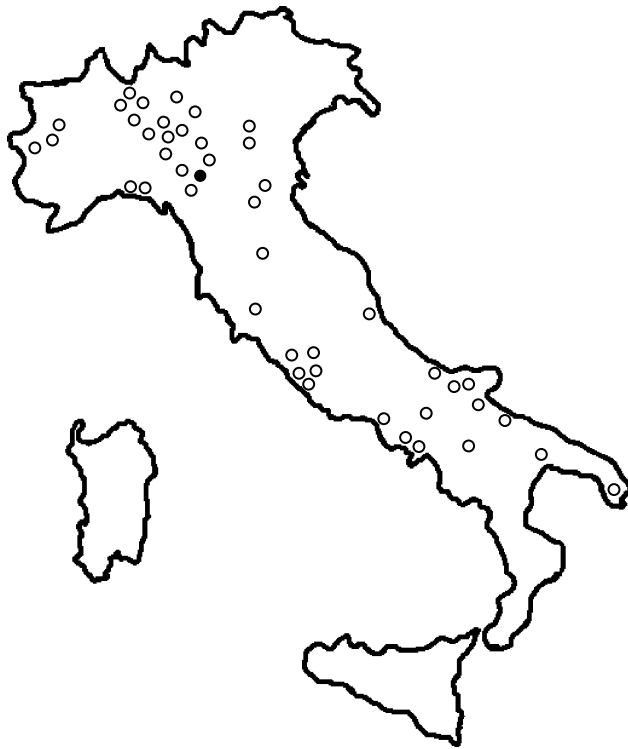
Figure 3   Distribution of Clients in Italy (Instance F01)

Italy are handled through a fleet of privately owned vehicles. These carriers are paid by mileage, so the company is interested in minimizing the total distance traveled. A sample instance (F01 in Table 3) is depicted in Figure 3; the filled circle represents the depot, and the empty circles represent the clients.

The demands consist of three-dimensional rectangular items that, once delivered, are unpacked and assembled to obtain the pieces of furniture. The vehicles are identical and have containers of standard ISO dimensions. To reduce the real problem to 3L-CVRP, we dropped some constraints. In particular, we did not consider time windows and pickup of damaged furniture at the customers' sites, and we approximated the real distances with Euclidean distances. Typical solutions include both single-day tours and multiple-day tours.

Concerning dimensions, the volumes of the items are between 1% and 4% of the vehicle volume, while the height of each item is between 10% and 50% of the vehicle height. The average total weight and total volume demanded by a client are, respectively, 0.7% and 5% of the corresponding vehicle capacities.

In Table 3 we show the results obtained by applying the tabu search algorithm of §4 to five instances provided by the company. For each instance, the algorithm was run three times, with different time limits: 1 hour, 10 hours, and 1 day. The substantial increase we needed for the time limit, with respect

to those adopted in Tables 1 and 2, was mainly due to the increased difficulty associated with the three-dimensional loading. Indeed, the excess of length was computed as in §5.1, but we allowed TS_{3L-SV} to perform up to 10 iterations (instead of 3). The first five entries give the instance name, the number of clients (n), the number of items (M), the number of vehicles (v), and the solution value (in kilometers) found by the initial heuristics (z_0). The three next pairs of entries give, for each time limit, the solution value (z) and the elapsed CPU time required to obtain this value (sec_z).

The initial heuristics find a feasible solution for all five instances, with an average solution value equal to 7,091 km. The three runs decrease this value by 33.7%, 35.3%, and 37.8%, respectively. For three instances the values obtained after 10 hours remain unchanged after 24 hours. When used in an operational planning situation, the preferred time limits are 10 hours (overnight) or 1 hour (when relatively fast decisions have to be taken).

5.3. Robustness and Parameters Setting

The parameters setting was performed on the 27 instances considered in §5.1. We start with the setting of the parameters needed to compute the score of a move (see (1)). The algorithm proved to be quite robust with respect to the starting values assigned to parameters α and β. For each tentative value T, we set $\alpha = T/L$ and $\beta = T/D$. The attempted values for T were 1, $\bar{c}$, $10\bar{c}$, $20\bar{c}$, and $30\bar{c}$, where $\bar{c}$ is the average edge cost. The value $T = 20\bar{c}$ produced slightly better results and was selected for the final configuration. Also changing the level of variation of α and β at each iteration did not produce significant variations. The diversification parameter γ was set to $\sqrt{2nv}$. This value is better than $\sqrt{nv}$, $\sqrt{3nv}$, and $\sqrt{4nv}$, which produced slightly worse results.

The algorithm proved to be more sensitive to the neighborhood size p and the tabu list tenure ϑ (see §4). Parameter p was set equal to $\min\{n/4, 20\}$, while worse results were produced by $\min\{n/6, 10\}$, $\min\{n/5, 15\}$, and $\min\{n/3, 25\}$. Parameter ϑ took instead the value $\min\{n/10, 15\}$, leading to better results than $\min\{n/2, 50\}$, $\min\{n/4, 50\}$, $\min\{n/6, 50\}$, $\min\{n/8, 25\}$, $\min\{n/10, 20\}$, and $\min\{n/10, 10\}$.

In the multistart approach, the algorithm proved to be sensitive to the maximum number of iterations given to each start. The final configuration (i.e., 25,000 iterations when $n \leq 25$, 5,000 when $25 < n < 50$, and 1,000 iterations when $n \geq 50$) was obtained by testing different values between 500 and 30,000 iterations. Concerning the single-start approach, a CPU time limit of 7,200 seconds for all the instances improved the solution value by 0.6% for the first group of nine instances and by 1% for the second group of nine

instances, while the average sec_z value increased from 2,058.9 to 2,778. Larger time limits did not produce significant improvements.

Concerning the number of calls to the inner tabu search $TS_{3L\text{-}SV}$, all values from 1 to 10 were attempted, with value 3 turning out to be the clear winner for the instances from the literature. In contrast, for the real-world instances this value did not lead to sufficiently good packings and was thus increased to 10.

6. Conclusions

We have introduced a new problem (denoted as 3L-CVRP) derived from the combination of the CVRP and the three-dimensional container-loading problem. The problem is very complex and is characterized by the presence of additional constraints, typical of freight transportation. We have developed a tabu search algorithm for the three-dimensional loading and included it in a tabu search algorithm for the overall problem. The resulting approach was tested on instances from the literature and on real-world instances, yielding interesting results.

References

Baker, B. S., E. G. Coffman, Jr., R. L. Rivest. 1980. Orthogonal packing in two dimensions. *SIAM J. Comput.* **9** 846–855.

Bortfeldt, D., H. Gehring. 2001. A hybrid genetic algorithm for the container loading problem. *Eur. J. Oper. Res.* **131** 143–161.

Christofides, N., C. Whitlock. 1977. An algorithm for two-dimensional cutting problems. *Oper. Res.* **25** 30–44.

Clarke, G., J. W. Wright. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.* **12** 568–581.

Cordeau, J.-F., G. Laporte. 2004. Tabu search heuristics for the vehicle routing problem. C. Rego, B. Alidaee, eds. *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search.* Kluwer, Boston, 145–163.

Cordeau, J.-F., M. Gendreau, G. Laporte. 1997. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks* **30** 105–119.

Cordeau, J.-F., M. Gendreau, A. Hertz, G. Laporte, J.-S. Sormany. 2005. New heuristics for the vehicle routing problem. A. Langevin, D. Riopel, eds. *Logistics Systems: Design and Optimization.* Springer, New York, 279–297.

Eley, M. 2002. Solving container loading problems by block arrangement. *Eur. J. Oper. Res.* **141** 393–409.

Fukasawa, R., H. Longo, J. L. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, F. Werneck. 2004. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Proc. X IPCO,* Vol. 3064. Springer Lecture Notes in Computer Science, New York, 1–15.

Gendreau, M., A. Hertz, G. Laporte. 1992. New insertion and postoptimization procedures for the traveling salesman problem. *Oper. Res.* **40** 1086–1094.

Gendreau, M., A. Hertz, G. Laporte. 1994. A tabu search heuristic for the vehicle routing problem. *Management Sci.* **40** 1276–1290.

Gendreau, M., M. Iori, G. Laporte, S. Martello. 2006. A tabu search approach to vehicle routing problems with two-dimensional loading constraints. *Networks.* Forthcoming.

Iori, M., J. J. Salazar González, D. Vigo. 2003. An exact approach for the symmetric capacitated vehicle routing problem with two dimensional loading constraints. Technical Report OR/03/04, DEIS, Università di Bologna, Bologna, Italy.

Levitin, K., R. Abezgaouz. 2003. Optimal routing of multiple-load AGV subject to LIFO loading constraints. *Comput. Oper. Res.* **30** 397–410.

Lodi, A. 2002. Multi-dimensional packing by tabu search. *Studia Informatica Universalis* **2** 111–126.

Lodi, A., S. Martello, D. Vigo. 1999. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS J. Comput.* **11** 345–357.

Martello, S., D. Pisinger, D. Vigo. 2000. The three-dimensional bin packing problem. *Oper. Res.* **48** 256–267.

Pisinger, D. 2002. Heuristics for the container loading problem. *Eur. J. Oper. Res.* **141** 382–392.

Reimann, M., K. Doerner, R. F. Hartl. 2004. D-ants: Savings based ants divide and conquer the vehicle routing problem. *Comput. Oper. Res.* **31** 563–591.

Toth, P., D. Vigo. 2002. *The Vehicle Routing Problem.* SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA.

Xu, H., Z.-L. Chen, S. Rajagopal, S. Arunapuram. 2003. Solving a practical pickup and delivery problem. *Transportation Sci.* **37** 347–364.