An Incremental Tabu Search Heuristic for the Generalized Vehicle Routing Problem with Time Windows

Luigi Moccia
Jean-François Cordeau
Gilbert Laporte

March 2010

CIRRELT-2010-12

An Incremental Tabu Search Heuristic for the Generalized Vehicle Routing Problem with Time Windows

Luigi Moccia[1], Jean-François Cordeau[2,3,*], Gilbert Laporte[2,4]

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Richerche, Via P. Bucci 41C, 87036 Rende (CS), Italy
[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)
[3] Canada Research Chair in Logistics and Transportation, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7
[4] Canada Research Chair in Distribution Management, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

Abstract. This paper describes an incremental neighbourhood tabu search heuristic for the generalized vehicle routing problem with time windows. The purpose of this work is to offer a general tool that can successfully be applied to a large variety of specific problems. The algorithm builds upon a previously developed tabu search heuristic by replacing its neighbourhood structure. The new neighbourhood is exponential in size, but the proposed evaluation procedure has polynomial complexity. Computational results are presented and demonstrate the effectiveness of the proposed approach.

Keywords. Generalized vehicle routing problems, time windows, tabu search, large neighbourhood search.

Acknowledgements. This work was partly supported by the MUR (Italy) under project LOGIN, and by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 227837-00 and 39682-05. This support is gratefully acknowledged. We thank Stefan Ropke for providing the GVRP instances, and for fruitful discussions.

* Corresponding author: Jean-Francois.Cordeau@cirrelt.ca

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2010

Introduction

The purpose of this paper is to describe an incremental tabu search heuristic for the generalized vehicle routing problem with time windows (GVRPTW) defined as follows. Let $G = (V, A)$ be a directed graph, where $V = \{0, 1, ..., n\}$ is the vertex set and $A = \{(i, j) : i, j \in V, i \neq j\}$ is the arc set. Vertex 0 is the depot at which is based a fleet of m identical vehicles, and the vertex set $V \setminus \{0\}$ represents customer sites. The set $V \setminus \{0\}$ is the union of p usually disjoint clusters $V_1, ..., V_p$. Cluster V_h represents the sites of customer h, and the vertices in V_h are characterised by the following customer attributes: a non-negative load q_h, a non-negative service duration d_h, and a time window $[e_h, l_h]$. A travel cost matrix c_{ij} and a travel time matrix t_{ij} are defined on A. The GVRPTW consists in designing at most m routes on G such that: (i) every route starts and ends at the depot; (ii) exactly one site per customer is visited; (iii) the total load and duration of a route r do not exceed Q_r and D_r, respectively; (iv) service at customer h begins within the interval $[e_h, l_h]$; (v) every vehicle leaves the depot and returns to the depot within the interval $[e_0, l_0]$; and (vi) the total routing cost is minimised. The GVRPTW reduces to the classical vehicle routing problem with time windows (VRPTW) when all clusters are singletons, and to the generalized traveling salesman problem with time windows (GTSPTW) when $m = 1$.

This model can be useful for a wide variety of applications, even when the time windows constraints are not considered: see Baldacci et al. (2010) for applications of the GVRP, and Laporte et al. (1996) for applications of the GTSP. Observe that a generalized version of a routing problem arises when customer visits can equivalently take place at more than one site. This gives rise to a location-routing problem because the sites to visit and the vehicle routes must be determined simultaneously (Laporte, 1988).

To the best of our knowledge, there is no literature on the GVRPTW. The closest study, by Bektaş et al. (2009), introduces a branch-and-cut algorithm and an adaptive large neighbourhood search algorithm for the GVRP. In previous articles, Cordeau et al. (2001, 2004) and Cordeau and Laporte (2001) have introduced a unified tabu search (UTS) heuristic which was successfully applied to the VRPTW and several of its extensions such as the multi-depot VRPTW (MDVRPTW), the periodic VRPTW (PVRPTW), and the site-dependent VRPTW (SDVRPTW). The flexibility of UTS has motivated us to adapt it to generalized vehicle routing problems.

With respect to the existing literature, this paper makes three main contributions. First, it introduces an effective heuristic algorithm for a new problem. Second, it describes how a previous heuristic for vehicle routing problems can be efficiently extended to handle generalized versions of the same problems. A key feature of our heuristic is the use of an incremental procedure to compute successive neighbourhoods. Third, it demonstrates through extensive computational experiments that the proposed general algorithm is competitive with specialised ones.

The remainder of the paper is organised as follows. The proposed algorithm, called incremental tabu search (ITS), is presented in the next section and assessed through extensive computational experiments. Finally, we report some conclusions.

The incremental tabu search algorithm

The ITS heuristic applies the same tabu search mechanisms as UTS but uses a new neighbourhood structure, which can be regarded as a generalisation of the one used in UTS. In the following, we first synthesise the UTS heuristic and its neighbourhood structure, and we then describe the new neighbourhood used in ITS. We will use for the VRPTW the notation defined in the introduction, and we note that $p = n$ for the VRPTW.

The unified tabu search heuristic and its neighbourhood structure

The UTS heuristic moves at each iteration from a solution x to another solution in its neighbourhood $N(x)$. In the following we denote as a solution a set of m routes starting and ending at the depot, and such that each customer is visited by one and only one vehicle. Hence, UTS considers a solution space where time window, duration, and capacity constraints may be violated. These constraints are handled in the objective function by means of penalty terms equal to the infeasibility value multiplied by a self-adjusting coefficient. If a solution is feasible with respect to any of these constraints, then the corresponding penalty is zero. The penalty coefficients are dynamically adjusted to produce a mix of feasible and infeasible solutions. Thus this relaxation mechanism facilitates the exploration of the search space and is particularly useful for tightly constrained instances. When customer i is removed from route r, its reinsertion in that route is forbidden for the next θ iterations, where θ is a parameter expressing the length of the tabu tenure. However, through an aspiration criterion, a forbidden reinsertion can be performed if this would allow reaching a solution of smaller cost than that of the best known solution containing customer i in route r. A continuous diversification mechanism is also implemented in UTS. The UTS algorithm performs a preset number of iterations η, and returns the best known feasible solution.

In the following we analyse the computational complexity of the UTS neighbourhood structure. The neighbourhood $N(x)$ consists of the solutions that can be obtained by moving each customer from its current route in x to another route. When evaluating the insertion of a customer in a different route all the positions in the new route are considered. Thus a full evaluation of the neighbourhood $N(x)$ computes p customer removals and $p(m-1)$ insertions. If we index a route by $r \in \{1, ..., m\}$, and we express as $C_r(x)$ the set of customers belonging to route r in solution x, then the insertion of a customer in route r requires the assessment of $|C_r(x)| + 1$ new customer sequences. Therefore, the $p(m-1)$ customer route insertions result in the evaluation of $\sum_{r=1}^{m}(|C_r(x)| + 1) \cdot (p - |C_r(x)|)$ customer sequences. Denoting by λ the number of customer sequences to be evaluated because of removal and insertions of customers, we obtain $\lambda = p + \sum_{r=1}^{m}(|C_r(x)| + 1) \cdot (p - |C_r(x)|)$. This leads to the following characterisation of the complexity of fully evaluating $N(x)$.

Proposition 1 *In the worst case, the full evaluation of the neighbourhood $N(x)$ has a computational cost of $O(n^2)$, and this occurs whenever the number of customers per route in the solution x is constant among the routes, hence equal to p/m.*

Proof — We can rewrite λ as $p + \sum_{r=1}^{m}(p|C_r(x)| - |C_r(x)|^2 + p - |C_r(x)|)$, and since

$\sum_{r=1}^{m} |C_r(x)| = p$, we have $\lambda = p^2 + pm - \sum_{r=1}^{m} |C_r(x)|^2$. By applying optimality conditions in the last expression, the term $-\sum_{r=1}^{m} |C_r(x)|^2$, under the constraint $\sum_{r=1}^{m} |C_r(x)| = p$, takes its maximal value $-p^2/m$ whenever $|C_r(x)| = p/m, \forall r \in \{1, ..., m\}$. Recalling that $p = n$ in a standard vehicle routing problem proves the proposition. $\square$

The *full* evaluation of the neighbourhood $N(x)$ is necessary only at the first iteration when the algorithm starts from an initial solution x_0. In fact, when evaluating $N(x_t)$, where x_t is the solution at the iteration $t \geq 1$, we can take advantage of the evaluation of the previous neighbourhood $N(x_{t-1})$. As described above, a solution x_t is obtained by moving a customer from its route in x_{t-1} to a new route in x_t. The impact of removing or inserting customers in the unchanged routes is the same in $N(x_{t-1})$ as it is in $N(x_t)$. Therefore, the evaluation of $N(x_t)$ requires assessing the impact of customer removals and insertions only in the two modified routes. We define this as an *incremental* neighbourhood evaluation. It can be easily verified that the incremental neighbourhood evaluation does not modify the order of magnitude of the worst case computational effort. For this reason, in the following we will refer to the full neighbourhood evaluation of $N(x)$ when discussing the worst case complexity of evaluating the new neighbourhood structure in ITS.

The incremental tabu search neighbourhood structure

In the ITS heuristic the solution space is defined similarly to that of UTS, i.e. we allow violations of time window, duration, and capacity constraints. The new neighbourhood $N^{'}(x)$ consists of all the solutions that can be obtained by moving each customer from its current route to another route, *while considering all the possible site choices for the customers in the modified routes*. We observe that, because of this added feature, $N^{'}(x)$ has an exponential size. Nevertheless, we have devised a neighbourhood evaluation procedure having a polynomial complexity. The idea stems from a well-known result related to the GTSP: given the sequence of clusters in a route, the optimal vertex choice can be determined by shortest path computation in a layered graph; see Renaud and Boctor (1998), and Bontoux et al. (2009) for an application of this result to a memetic algorithm for the GTSP. Similarly, in the GVRP we can use a layered graph to optimally solve the site selection problem when the customer sequence is fixed in a route. Figure 1 illustrates how this layered graph is generated. We represent the customer sequence in a route by an ordered set $\pi = \{h, k, ..., z\}$, and we introduce a vertex $0^{'}$ corresponding to the departure from the depot, and a vertex $0^{''}$ which denotes the arrival at the depot. The vertex set of the layered graph G_π consists of the vertices $0^{'}$, $0^{''}$, and of the vertices in $\cup_{j\in\pi} V_j$. Vertex $0^{'}$ is linked by arcs only to the vertices corresponding to the sites of the first visited customer, which in our example are the vertices belonging to the set V_h. The cost of each of these arcs is equal to the cost between the depot and the corresponding customer sites, i.e. $c_{0^{'}i} = c_{0i}, \forall i \in V_h$. In turn, the vertices belonging to the set V_h are linked to vertices belonging to the set V_k, etc. These corresponding arcs have costs equal to those of the graph G. Finally, the vertices corresponding to the sites of the last visited customer, i.e. the vertices in V_z, are linked to the arrival vertex $0^{''}$ by arcs with costs $c_{j0^{''}} = c_{j0}, \forall j \in V_z$. The optimal site choice for a given customer

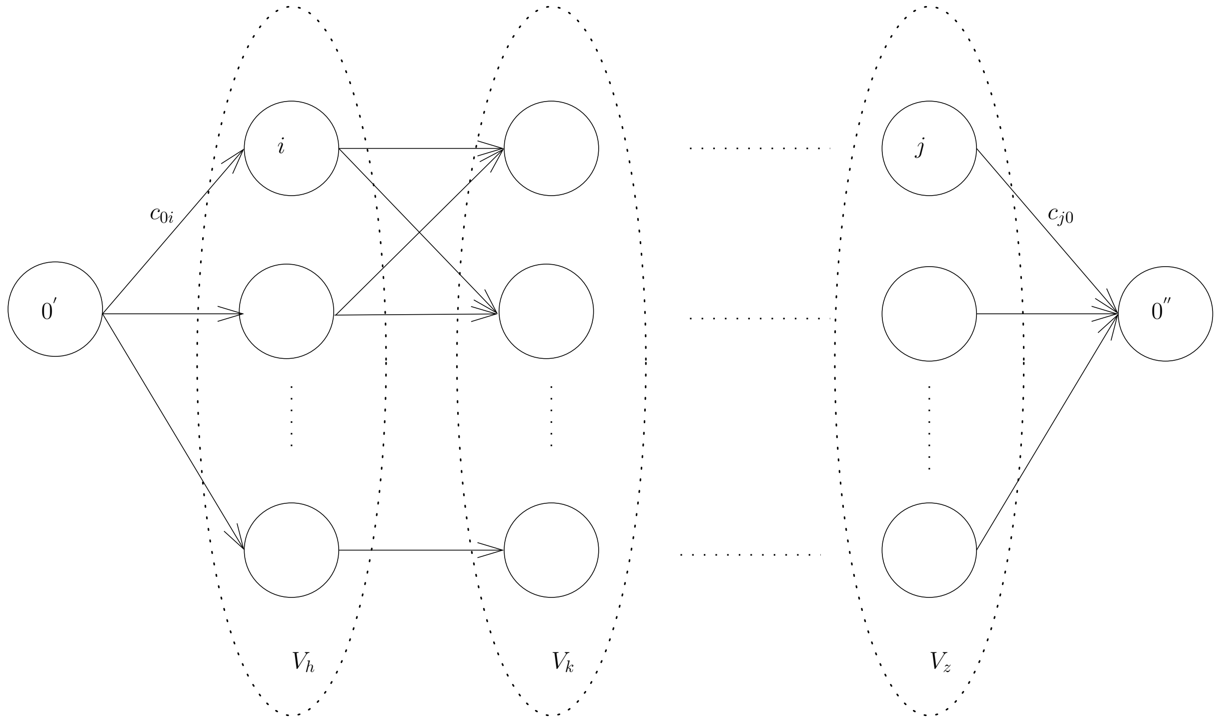

Figure 1: The layered graph G_π for the customer sequence $\pi = (h, k, ..., z)$

sequence π can readily be determined by computing a shortest path from $0^{'}$ to $0^{''}$ in G_π. The cost of this shortest path is the optimal cost of the route visiting the customer sequence π.

It is well known, see e.g. Ahuja et al. (1993), that a shortest path in an acyclic graph can be computed by the so-called reaching algorithm. This algorithm computes labels from the origin to the destination vertex following their topological order (which in a layered graph does not need to be computed because it can be trivially established). This algorithm has a complexity equal to the number of arcs of the graph. Observe that, analogously to the case described for $N(x)$, the full evaluation of the neighbourhood $N^{'}(x)$ would require the assessment of $O(p^2)$ removals and insertions of customers. Therefore, the reaching algorithm should compute $O(p^2)$ shortest paths. We have to establish worst case conditions for the computation of these shortest paths. Using arguments similar to those of Proposition 1, it can be determined that these worst case conditions occur whenever each route contains p/m customers, and there is a constant number n/p of vertices per cluster. These conditions result in a computational cost per shortest path by the reaching algorithm equal to $O(n^2/(mp))$, which leads to the following result.

Proposition 2 *In the worst case the full evaluation of the neighbourhood $N^{'}(x)$ has a computational cost of $O(p^2 \times n^2/(mp)) = O(n^2p/m)$ when using the reaching algorithm.*

This result indicates that the evaluation of $N^{'}(x)$ should be p/m times more computationally expensive than that of $N(x)$. However, we can significantly speed up computations by taking advantage of the shortest paths computed at previous steps of our algorithm. Indeed, in the exploration of the neighbourhood $N^{'}(x)$, the solutions encountered differ only marginally from the current solution x. The assessment of a solution $y \in N^{'}(x)$ can be regarded as an *incremental optimisation problem* in the sense of Şeref et al. (2009). In the following we show how to considerably reduce the computational effort of this problem. To this end we will employ bi-directional label setting and label recycling as in the work of Hu and Raidl (2008) for the GTSP 2-opt neighbourhood structure.

We define forward labels ϕ as those generated by the reaching algorithm. A label ϕ_i expresses the cost of a shortest path from $0^{'}$ to i in the layered graph. We can equivalently compute a shortest path from $0^{'}$ to $0^{''}$ by applying the reaching algorithm in reverse mode, i.e. new labels are expanded from $0^{''}$ to $0^{'}$ following their reverse topological order. We define these new labels as backward labels β. Thus, a label β_i indicates the cost of a shortest path from i to $0^{''}$ in the layered graph. By computing both types of labels we obtain the following property:

Property 1 *The cost of the shortest path c^* can be computed for each cluster V as the minimum of the sum of forward and backward labels at each vertex of the cluster, $c^* = \min_{i \in V}\{\phi_i + \beta_i\}$.*

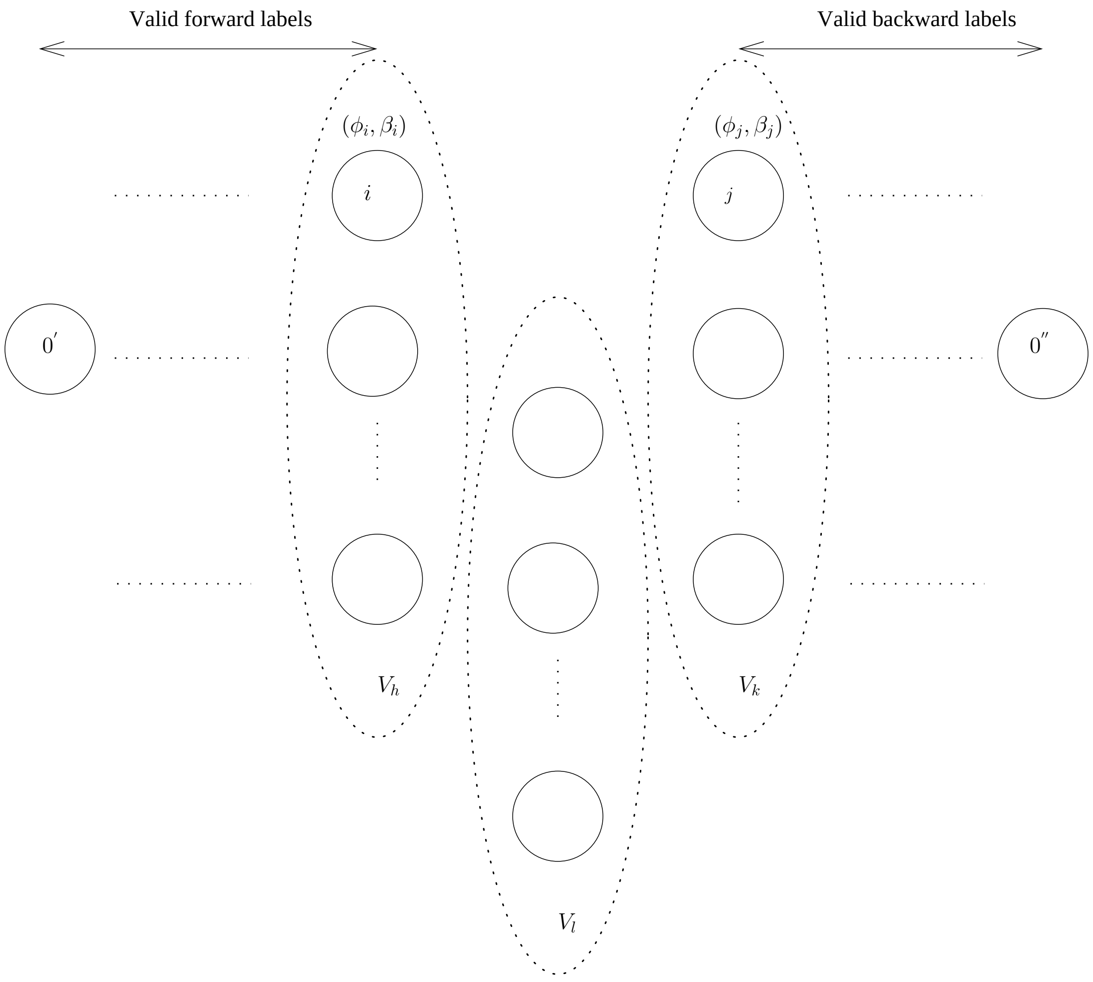

Figure 2: Valid labels when inserting a cluster in a route

As described above, we have to evaluate the insertion of a customer in each position of an existing route. Let $\mu = \{..., h, k, ...\}$ be a customer sequence in a given route. When evaluating the insertion of a customer l after customer h we need the cost of a shortest path from $0^{'}$ to $0^{''}$ in $G_{\mu^{'}}$, where $\mu^{'}$ is defined as $\{..., h, l, k, ...\}$, but it is not necessary to compute a shortest path afresh for each potential insertion. First observe that the forward labels generated by the shortest path computation for the customer sequence μ are still valid for the vertices of the layered graph from vertex $0^{'}$ to the vertices in the set V_h in $G_{\mu'}$. More formally, a label ϕ_i obtained by the shortest path computation in G_μ is valid for the shortest path to be computed in $G_{\mu'}$ if the value of the new forward label for the vertex i in $G_{\mu'}$ is equal to ϕ_i. The label is said to be invalid otherwise. Therefore, the valid labels need not be recomputed, and the reaching algorithm only computes the labels from the vertices in the set V_l to vertex $0^{''}$. Similarly to the previous case, we can recycle the valid backward labels, and when computing the shortest path backward we only compute the backward labels from the vertices in the set V_l to vertex $0^{'}$ (see Figure 2). However, we observe that in order to evaluate the impact of a customer insertion in a given position of a route, we do not have to fully re-evaluate invalid forward and backward labels. Recalling Property 1, we only require *one* cluster in the layered graph with the valid status for both types of labels. In fact, extending the forward labels from the vertices of the set V_h to the vertices of the set V_l and extending the backward labels from the vertices of the set V_k to the vertices of the set V_l suffices to produce a set of valid ϕ and β labels for cluster V_l in $G_{\mu'}$. Hence, the procedure just outlined exactly evaluates the impact of the insertion of customer l between consecutive customers in a route. Its computational burden is proportional to the number of arcs in the layered graph between the vertices of the sets V_h and V_l, and between the vertices of the sets V_l and V_k, i.e. it is equal to $O(|V_h| \cdot |V_l| + |V_l| \cdot |V_k|)$. Under worst case conditions introduced above (customers equally distributed along the routes, and a constant number of vertices per cluster), the insertion procedure with bi-directional label setting and label recycling has a computational cost of $O(2n^2/p^2)$. The procedure needed to evaluate the effect of a customer deletion has a similar computational cost. This result can be derived in way similar to the insertion case, and hence its proof is omitted. We then have the following result:

Proposition 3 *The incremental procedures just introduced bring the worst case computational cost of evaluating $N^{'}(x)$ down to $O(n^2)$.*

Because the complexity of evaluating $N(x)$ in UTS for the standard VRPTW is $O(n^2)$, this result for the GVRPTW indicates that the performance of ITS depends on the number of sites only.

Once a shortest path is computed, we have a choice of vertices for the given customer sequence. The evaluation of the full route is done as in the UTS heuristic, using the forward time slack concept described in Cordeau et al. (2004). We observe that label recycling applies to the shortest path computations only. In fact, a forward label indicates the cost of a shortest path from the origin to the vertex, and is not necessarily equal to the arrival time at the vertex (it can be if $c_{ij} = t_{ij}$, and if the time windows are not binding). The new neighbourhood structure for the GVRPTW can easily be extended to handle periodic, multi-depot, and site-dependent generalized vehicle routing problems, similarly to what is done for the VRPTW in Cordeau et al. (2001), and Cordeau and Laporte (2001).

Computational Results

We now present extensive computational experiments to assess the effectiveness of the ITS algorithm. We have first evaluated ITS on some GVRP instances presented in Bektaş et al. (2009). These instances offer a considerable advantage because tight lower bounds were obtained for them by the branch-and-cut algorithm of Bektaş et al. (2009). Furthermore, in the same article, these instances were solved by an adaptive large neighbourhood search (ALNS) algorithm which is a state-of-the-art metaheuristic. We first provide a comparison of ITS and ALNS on these GVRP instances.

In order to asses the ITS heuristic on instances with time related constraints we have used some MDVRPTW instances. As shown by Baldacci et al. (2010), the MDVRP can be modelled as a GVRP. The solution values obtained by ITS on these MDVRPTW instances were evaluated against results provided by the UTS algorithm. This comparison also provides clues regarding the advantage of a general algorithm versus a specialised one for a given class of problems.

Computational experiments on GVRP instances

The ITS algorithm was coded in ANSI C, and the computational experiments were performed on a desktop computer with an Intel Core Duo 1.83 GHz processor. Our implementation of ITS uses the following procedure to obtain a starting solution x_0: the m routes are initialised by inserting seed customers as distant as possible from each other and from the depot; then, the remaining customers are inserted into routes in a greedy fashion. This procedure was developed to initialise the ITS heuristic in a way similar to that followed in the ALNS algorithm of Bektaş et al. (2009). To evaluate performance of ITS we have used the GVRP instances of Bektaş et al. (2009). In these instances the average number of vertices per cluster is equal to two or three. We refer to Bektaş et al. (2009) for additional details regarding these instances. The ITS heuristic optimally chooses the customer sites given the customer sequence, whereas ALNS does not possess such a feature. We therefore conjecture that ITS should perform better on instances with higher values of n/p, the average number of vertices per cluster in our notation. As we show in the following, the computational experiments reported in Tables 1, 2, 3, and 4, support this conjecture.

These tables are organised as follows. We report in the first column the instance codes as in Bektaş et al. (2009). The second column lists the lower bounds obtained by the branch-and-cut algorithm of these authors. Columns 3 to 5 report the results of their ALNS algorithm, and the last three columns are for the ITS results. We report a solution quality index computed by scaling the heuristic solution value (UB) to the lower bound (LB), where a value of 100.00 corresponds to the lower bound. Thus, an index value of 100.00 indicates an optimal solution. Average solution quality indexes for ALNS and ITS are reported in the last row of the tables.

Computational experiments on small to medium instances with $n/p = 2$ are presented in Table 1. The ALNS algorithm obtains a slightly better average solution quality on these instances. However, the ITS algorithm is slightly better on small to medium instances with $n/p = 3$, see Table 2. As can be seen, ITS consistently reaches optimal or near-optimal solutions. These trends are similar for larger instances, see Tables 3 and 4. On these more difficult instances, the average deviation from the lower bound is just above 5%. We observe that the computational times of the two algorithms are not directly comparable because of different computer processors (Bektaş et al. (2009) used a faster machine). Now, the running time of ITS is proportional to its number η of iterations. Since $\eta = 10^5$ seems necessary to sufficiently explore the solution space, this heuristic seems to be slower than ALNS, even when accounting for the differences in machine speeds.

The comparison presented above was intended to be the toughest possible for ITS, because it is equivalent to testing the UTS algorithm on VRP instances, which is not what UTS was designed for. In fact, GVRP instances do not allow the use of ITS at its full strength because ITS, derived from UTS, works best on routing problems with time windows, and maximal route duration. Still, ITS produces high quality solutions on GVRP instances, and the results obtained on instances with a larger n/p value indicate that this algorithm, although slower than ALNS, is competitive in terms of solution quality with a state-of-the-art metaheuristic expressly designed for the GVRP.

Computational experiments on MDVRPTW instances

As observed by Baldacci et al. (2010), the MDVRP can be modelled as a GVRP on an expanded graph where each customer is represented by a cluster of vertices, one for each depot. This graph contains customer-depot vertices indicated by pairs (h, d), where h is the index of a customer, and d is the index of a depot. An arc connecting two vertices (h, d) and (k, d) related to the same depot d has a cost equal to the cost of the arc between the customers h and k in the original graph. Arcs between vertices related to different depots have a cost equal to a sufficiently large value. The customer-depot vertices are connected to an artificial depot vertex. The cost of an arc connecting the artificial depot to a customer-depot vertex (h, d) is equal to the cost of the arc connecting depot d to customer h in the original graph. This reformulation can also be used to transform a MDVRPTW in a GVRPTW by assigning the time window of each customer h to the corresponding customer-depot vertices (h, d). We observe that this reformulation exactly transforms a MDVRPTW into a GVRPTW if there are no depot-specific constraints in the MDVRPTW. For example, if there are constraints such as upper bounds on the number of vehicles available at the depots, these constraints do not translate to the standard GVRPTW where an upper bound upon the number of available vehicles can be set for *all* routes. For similar reasons, in the MDVRPTW all depots must have the same time window.

The problem reformulation just described has the drawback of considerably increasing the size of the GVRPTW graph with respect to the size of the original MDVRPTW graph. Observe that the majority of the arcs in the expanded graph, those with an artificial large cost, will never be part of an optimal solution. Thus, testing the ITS algorithm on MDVRPTW instances modelled as GVRPTW poses a remarkable challenge. Still, this reformulation allow us to compare the ITS results with those of UTS. The UTS algorithm is in fact well known for providing high quality solutions on MDVRPTW instances.

Both the ITS and the UTS algorithms were tested on the computer platform used for the previous tests. In this implementation of the ITS algorithm we have used a sweep algorithm to generate an initial solution, which is the default starting procedure of UTS. The instances were derived from those of Cordeau et al. (2001). These new instances differ from those of Cordeau et al. (2001) by having a very large value for the maximum number of routes per depot. This modification was necessary in order to have equivalent instances for both the MDVRPTW and the GVRPTW reformulation. The computational results of UTS and ITS on these MDVRPTW instances are reported in Table 5. Here, we compute the solution quality index by comparing the ITS solution value to the best known solution value. The average solution quality provided by ITS is similar to the one provided by UTS. The computational times of ITS are larger than the ones of UTS, but they remain in the same order of magnitude, in spite of the significant increase in size of the underlying graph. Observe that in these instances the number of depots varies between four and six (third column of Table 5).

These results indicate that a general tool as ITS can be successfully applied to the solution of specific problems. On the test instances, it consistently yields optimal or near-optimal solutions. Because of the large variety of different location-routing problems that can be modelled as GVRPTW, this is, in our opinion, a noteworthy advantage.

Conclusions

We have presented a new incremental tabu search heuristic, called ITS, for the generalized vehicle routing problems with time windows. The algorithm builds upon a previously developed one by replacing its neighbourhood structure. Although the new neighbourhood is exponential in size, we have presented an evaluation procedure which is polynomial. Computational experiments were performed to assess the effectiveness of the proposed approach. The algorithm provides optimal or near-optimal solutions on instances for which lower bounds are available. In this respect, ITS compares favourably to state-of-the-art metaheuristics.

Acknowledgements

This work was partly supported by the MUR (Italy) under project LOGIN, and by the Natural Sciences and Engineering Council of Canada under grants 227837-00 and 39682-05. This support is gratefully acknowledged. We thank Stefan Ropke for providing the GVRP instances, and for fruitful discussions.

References

Ahuja, R. K., Magnanti, T. L., and Orlin, J. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, New Jersey.

Baldacci, R., Bartolini, E., and Laporte, G. (2010). Some applications of the generalized vehicle routing problem. *Journal of the Operational Research Society*. Forthcoming.

Bektaş, T., Erdoğan, G., and Ropke, S. (2009). Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. Technical report. http://www.diku.dk/hjemmesider/ansatte/sropke.

Bontoux, B., Artigues, C., and Feillet, D. (2009). A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem. *Computers & Operations Research*, In Press, Corrected Proof, Available online 18 May 2009.

Cordeau, J.-F. and Laporte, G. (2001). A tabu search algorithm for the site dependent vehicle routing problem with time windows. *INFOR*, 39(3):292–298.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2004). Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society*, 55(5):542 – 546.

Hu, B. and Raidl, G. (2008). Effective neighborhood structures for the generalized traveling salesman problem. In van Hemert, J. and Cotta, C., editors, *Evolutionary Computation in Combinatorial Optimisation*, volume 4972 of *Lecture Notes in Computer Science*, pages 36–47. Springer, Berlin.

Laporte, G. (1988). Location-routing problems. In Golden, B. L. and Assad, A. A., editors, *Vehicle Routing: Methods and Studies*, pages 163–197. North-Holland, Amsterdam.

Laporte, G., Asef-Vaziri, A., and Sriskandarajah, C. (1996). Some applications of the generalized travelling salesman problem. *Journal of the Operational Research Society*, 47(12):1461–1467.

Renaud, J. and Boctor, F. F. (1998). An efficient composite heuristic for the symmetric generalized traveling salesman problem. *European Journal of Operational Research*, 108(3):571 – 584.

Şeref, O., Ahuja, R. K., and Orlin, J. B. (2009). Incremental network optimization: Theory and algorithms. *Operations Research*, 57(3):586–594.

			ALNS			ITS	
Instance	LB	UB	100 × UB/LB	Time (sec.)	UB	100 × UB/LB	Time (sec.)
A-n32-k5-C16-V2	519.0	519	**100.00**	0.3	519	**100.00**	9.5
A-n33-k5-C17-V3	451.0	451	**100.00**	0.3	451	**100.00**	8.8
A-n33-k6-C17-V3	465.0	465	**100.00**	0.3	465	**100.00**	9.0
A-n34-k5-C17-V3	489.0	489	**100.00**	0.3	489	**100.00**	9.2
A-n36-k5-C18-V2	505.0	505	**100.00**	0.3	505	**100.00**	12.0
A-n37-k5-C19-V3	432.0	432	**100.00**	0.4	432	**100.00**	11.1
A-n37-k6-C19-V3	584.0	584	**100.00**	0.3	584	**100.00**	11.1
A-n38-k5-C19-V3	476.0	476	**100.00**	0.3	476	**100.00**	11.0
A-n39-k5-C20-V3	557.0	557	**100.00**	0.4	557	**100.00**	12.5
A-n39-k6-C20-V3	544.0	544	**100.00**	0.4	544	**100.00**	12.6
A-n44-k6-C22-V3	608.0	608	**100.00**	0.4	608	**100.00**	15.1
A-n45-k6-C23-V4	613.0	613	**100.00**	0.4	613	**100.00**	13.7
A-n45-k7-C23-V4	674.0	674	**100.00**	0.4	674	**100.00**	13.7
A-n46-k7-C23-V4	593.0	593	**100.00**	0.4	593	**100.00**	13.5
A-n48-k7-C24-V4	667.0	667	**100.00**	0.4	667	**100.00**	15.4
A-n53-k7-C27-V4	603.0	603	**100.00**	0.5	603	**100.00**	19.4
A-n54-k7-C27-V4	690.0	690	**100.00**	0.5	690	**100.00**	18.7
A-n55-k9-C28-V5	699.0	699	**100.00**	0.5	699	**100.00**	16.7
A-n60-k9-C30-V5	769.0	769	**100.00**	0.6	769	**100.00**	19.3
A-n61-k9-C31-V5	638.0	638	**100.00**	0.6	638	**100.00**	20.7
A-n62-k8-C31-V4	740.0	740	**100.00**	0.6	740	**100.00**	26.0
A-n63-k10-C32-V5	801.0	801	**100.00**	0.6	801	**100.00**	22.7
A-n63-k9-C32-V5	900.3	912	**101.30**	0.6	912	**101.30**	21.9
A-n64-k9-C32-V5	763.0	763	**100.00**	0.6	763	**100.00**	23.0
A-n65-k9-C33-V5	682.0	682	**100.00**	0.7	682	**100.00**	23.3
A-n69-k9-C35-V5	680.0	680	**100.00**	0.8	680	**100.00**	26.6
A-n80-k10-C40-V5	957.4	997	**104.14**	1.0	997	**104.14**	35.5
B-n31-k5-C16-V3	441.0	441	**100.00**	0.3	441	**100.00**	8.2
B-n34-k5-C17-V3	472.0	472	**100.00**	0.3	472	**100.00**	9.3
B-n35-k5-C18-V3	626.0	626	**100.00**	0.3	626	**100.00**	9.8
B-n38-k6-C19-V3	451.0	451	**100.00**	0.3	451	**100.00**	10.7
B-n39-k5-C20-V3	357.0	357	**100.00**	0.4	357	**100.00**	11.1
B-n41-k6-C21-V3	481.0	481	**100.00**	0.4	481	**100.00**	13.5
B-n43-k6-C22-V3	483.0	483	**100.00**	0.4	483	**100.00**	14.8
B-n44-k7-C22-V4	540.0	540	**100.00**	0.4	540	**100.00**	12.9
B-n45-k5-C23-V3	497.0	497	**100.00**	0.5	497	**100.00**	16.0
B-n45-k6-C23-V4	478.0	478	**100.00**	0.4	478	**100.00**	13.8
B-n50-k7-C25-V4	449.0	449	**100.00**	0.5	449	**100.00**	16.4
B-n50-k8-C25-V5	916.0	916	**100.00**	0.5	916	**100.00**	13.9
B-n51-k7-C26-V4	651.0	651	**100.00**	0.5	651	**100.00**	15.1
B-n52-k7-C26-V4	450.0	450	**100.00**	0.5	450	**100.00**	16.3
B-n56-k7-C28-V4	486.0	486	**100.00**	0.6	492	101.23	16.4
B-n57-k7-C29-V4	751.0	751	**100.00**	0.5	751	**100.00**	21.5
B-n57-k9-C29-V5	942.0	942	**100.00**	0.5	942	**100.00**	18.5
B-n63-k10-C32-V5	816.0	816	**100.00**	0.6	816	**100.00**	23.3
B-n64-k9-C32-V5	509.0	509	**100.00**	0.7	509	**100.00**	21.8
B-n66-k9-C33-V5	808.0	808	**100.00**	0.7	808	**100.00**	24.6
B-n67-k10-C34-V5	673.0	673	**100.00**	0.7	673	**100.00**	24.8
B-n68-k9-C34-V5	704.0	704	**100.00**	0.7	704	**100.00**	25.7
B-n78-k10-C39-V5	803.0	803	**100.00**	0.8	804	100.12	34.1
P-n16-k8-C8-V5	239.0	239	**100.00**	0.2	239	**100.00**	2.2
P-n19-k2-C10-V2	147.0	147	**100.00**	0.2	147	**100.00**	1.7
P-n20-k2-C10-V2	154.0	154	**100.00**	0.2	154	**100.00**	1.7
P-n21-k2-C11-V2	160.0	160	**100.00**	0.2	162	101.25	1.9
P-n22-k2-C11-V2	162.0	162	**100.00**	0.2	163	100.62	1.9
P-n22-k8-C11-V5	314.0	314	**100.00**	0.2	314	**100.00**	3.2
P-n23-k8-C12-V5	312.0	312	**100.00**	0.2	312	**100.00**	3.7
P-n40-k5-C20-V3	294.0	294	**100.00**	0.4	294	**100.00**	12.6
P-n45-k5-C23-V3	337.0	337	**100.00**	0.5	337	**100.00**	15.9
P-n50-k10-C25-V5	410.0	410	**100.00**	0.5	410	**100.00**	14.3
P-n50-k7-C25-V4	353.0	353	**100.00**	0.5	353	**100.00**	16.1
P-n50-k8-C25-V4	378.4	392	**103.60**	0.5	421	111.26	16.7
P-n51-k10-C26-V6	427.0	427	**100.00**	0.5	427	**100.00**	12.4
P-n55-k10-C28-V5	415.0	415	**100.00**	0.5	415	**100.00**	17.3
P-n55-k15-C28-V8	545.3	555	**101.78**	0.5	565	103.61	13.3
P-n55-k7-C28-V4	361.0	361	**100.00**	0.6	361	**100.00**	20.0
P-n55-k8-C28-V4	361.0	361	**100.00**	0.6	361	**100.00**	20.0
P-n60-k10-C30-V5	433.0	443	**102.30**	0.6	443	**102.30**	20.0
P-n60-k15-C30-V8	553.9	565	**102.01**	0.6	565	**102.01**	13.7
P-n65-k10-C33-V5	487.0	487	**100.00**	0.7	487	**100.00**	24.1
P-n70-k10-C35-V5	485.0	485	**100.00**	0.8	485	**100.00**	27.3
P-n76-k4-C38-V2	383.0	383	**100.00**	1.0	383	**100.00**	58.1
P-n76-k5-C38-V3	405.0	405	**100.00**	0.9	405	**100.00**	45.7
P-n101-k4-C51-V2	455.0	455	**100.00**	1.9	455	**100.00**	119.1
Average			**100.20**			*100.38*	

Table 1: Computational results on GVRP small to medium instances with $n/p = 2$. Bold entries correspond to the best solution quality index for each row.

			ALNS			ITS	
Instance	LB	UB	100 $\times$ UB/LB	Time (sec.)	UB	100 $\times$ UB/LB	Time (sec.)
A-n32-k5-C11-V2	386.0	386	**100.00**	0.2	386	**100.00**	4.3
A-n33-k5-C11-V2	315.0	318	100.95	0.2	315	**100.00**	4.6
A-n33-k6-C11-V2	370.0	370	**100.00**	0.2	370	**100.00**	5.7
A-n34-k5-C12-V2	419.0	419	**100.00**	0.2	419	**100.00**	6.0
A-n36-k5-C12-V2	396.0	396	**100.00**	0.2	396	**100.00**	5.0
A-n37-k5-C13-V2	347.0	347	**100.00**	0.2	347	**100.00**	6.1
A-n37-k6-C13-V2	431.0	431	**100.00**	0.2	431	**100.00**	7.5
A-n38-k5-C13-V2	367.0	367	**100.00**	0.2	367	**100.00**	7.0
A-n39-k5-C13-V2	364.0	364	**100.00**	0.2	364	**100.00**	6.8
A-n39-k6-C13-V2	403.0	403	**100.00**	0.2	403	**100.00**	7.6
A-n44-k6-C15-V2	503.0	503	**100.00**	0.3	503	**100.00**	10.1
A-n45-k6-C15-V3	474.0	474	**100.00**	0.3	474	**100.00**	8.5
A-n45-k7-C15-V3	475.0	475	**100.00**	0.3	475	**100.00**	8.8
A-n46-k7-C16-V3	462.0	462	**100.00**	0.3	462	**100.00**	9.6
A-n48-k7-C16-V3	451.0	451	**100.00**	0.3	451	**100.00**	9.8
A-n53-k7-C18-V3	440.0	440	**100.00**	0.4	440	**100.00**	12.1
A-n54-k7-C18-V3	482.0	482	**100.00**	0.4	482	**100.00**	12.2
A-n55-k9-C19-V3	473.0	473	**100.00**	0.4	473	**100.00**	13.7
A-n60-k9-C20-V3	595.0	595	**100.00**	0.4	595	**100.00**	16.1
A-n61-k9-C21-V4	473.0	473	**100.00**	0.4	473	**100.00**	14.2
A-n62-k8-C21-V3	596.0	596	**100.00**	0.4	596	**100.00**	16.9
A-n63-k10-C21-V4	593.0	593	**100.00**	0.4	593	**100.00**	14.6
A-n63-k9-C21-V3	625.6	642	**102.62**	0.4	643	102.78	17.5
A-n64-k9-C22-V3	536.0	536	**100.00**	0.4	536	**100.00**	17.2
A-n65-k9-C22-V3	500.0	500	**100.00**	0.4	500	**100.00**	18.8
A-n69-k9-C23-V3	520.0	520	**100.00**	0.4	520	**100.00**	20.7
A-n80-k10-C27-V4	679.4	710	**104.50**	0.6	710	**104.50**	24.9
B-n31-k5-C11-V2	356.0	356	**100.00**	0.2	356	**100.00**	4.3
B-n34-k5-C12-V2	369.0	369	**100.00**	0.2	369	**100.00**	5.1
B-n35-k5-C12-V2	501.0	501	**100.00**	0.2	501	**100.00**	5.7
B-n38-k6-C13-V2	370.0	370	**100.00**	0.2	370	**100.00**	7.7
B-n39-k5-C13-V2	280.0	280	**100.00**	0.2	280	**100.00**	6.0
B-n41-k6-C14-V2	407.0	407	**100.00**	0.2	407	**100.00**	8.5
B-n43-k6-C15-V2	343.0	343	**100.00**	0.3	343	**100.00**	9.5
B-n44-k7-C15-V3	395.0	395	**100.00**	0.3	395	**100.00**	8.3
B-n45-k5-C15-V2	410.0	422	102.93	0.3	410	**100.00**	8.4
B-n45-k6-C15-V2	336.0	336	**100.00**	0.3	336	**100.00**	10.3
B-n50-k7-C17-V3	393.0	393	**100.00**	0.3	393	**100.00**	11.0
B-n50-k8-C17-V3	598.0	598	**100.00**	0.3	598	**100.00**	11.0
B-n51-k7-C17-V3	511.0	511	**100.00**	0.3	511	**100.00**	10.6
B-n52-k7-C18-V3	359.0	359	**100.00**	0.3	359	**100.00**	11.1
B-n56-k7-C19-V3	356.0	356	**100.00**	0.4	356	**100.00**	12.6
B-n57-k7-C19-V3	558.0	558	**100.00**	0.4	558	**100.00**	13.0
B-n57-k9-C19-V3	681.0	681	**100.00**	0.4	681	**100.00**	13.8
B-n63-k10-C21-V3	599.0	599	**100.00**	0.4	599	**100.00**	17.2
B-n64-k9-C22-V4	452.0	452	**100.00**	0.5	452	**100.00**	15.4
B-n66-k9-C22-V3	609.0	609	**100.00**	0.4	609	**100.00**	18.1
B-n67-k10-C23-V4	558.0	558	**100.00**	0.5	558	**100.00**	17.0
B-n68-k9-C23-V3	523.0	523	**100.00**	0.4	523	**100.00**	20.1
B-n78-k10-C26-V4	606.0	606	**100.00**	0.5	606	**100.00**	21.1
P-n16-k8-C6-V4	170.0	170	**100.00**	0.1	170	**100.00**	1.5
P-n19-k2-C7-V1	111.0	111	**100.00**	0.1	111	**100.00**	0.1
P-n20-k2-C7-V1	117.0	117	**100.00**	0.1	117	**100.00**	0.1
P-n21-k2-C7-V1	117.0	117	**100.00**	0.1	117	**100.00**	0.1
P-n22-k2-C8-V1	111.0	111	**100.00**	0.2	111	**100.00**	0.1
P-n22-k8-C8-V4	249.0	249	**100.00**	0.2	249	**100.00**	2.4
P-n23-k8-C8-V3	174.0	174	**100.00**	0.2	174	**100.00**	2.9
P-n40-k5-C14-V2	213.0	213	**100.00**	0.3	213	**100.00**	7.5
P-n45-k5-C15-V2	238.0	238	**100.00**	0.3	238	**100.00**	9.2
P-n50-k10-C17-V4	292.0	292	**100.00**	0.3	292	**100.00**	9.2
P-n50-k7-C17-V3	261.0	261	**100.00**	0.3	261	**100.00**	10.7
P-n50-k8-C17-V3	262.0	262	**100.00**	0.3	262	**100.00**	11.2
P-n51-k10-C17-V4	309.0	309	**100.00**	0.3	309	**100.00**	9.6
P-n55-k10-C19-V4	301.0	301	**100.00**	0.4	301	**100.00**	11.8
P-n55-k15-C19-V6	378.0	378	**100.00**	0.4	378	**100.00**	9.6
P-n55-k7-C19-V3	271.0	271	**100.00**	0.4	271	**100.00**	13.3
P-n55-k8-C19-V3	274.0	274	**100.00**	0.4	274	**100.00**	13.5
P-n60-k10-C20-V4	325.0	325	**100.00**	0.4	325	**100.00**	13.0
P-n60-k15-C20-V5	379.3	382	**100.73**	0.4	382	**100.73**	11.6
P-n65-k10-C22-V4	372.0	372	**100.00**	0.4	372	**100.00**	15.5
P-n70-k10-C24-V4	385.0	385	**100.00**	0.5	385	**100.00**	18.3
P-n76-k4-C26-V2	309.0	320	103.56	0.6	309	**100.00**	20.3
P-n76-k5-C26-V2	309.0	309	**100.00**	0.6	309	**100.00**	27.4
P-n101-k4-C34-V2	370.0	374	101.08	1.0	370	**100.00**	31.8
Average			*100.22*			**100.11**	

Table 2: Computational results on GVRP small to medium instances with $n/p = 3$. Bold entries correspond to the best solution quality index for each row.

		ALNS			ITS		
Instance	LB	UB	100 × UB/LB	Time (sec.)	UB	100 × UB/LB	Time (sec.)
M-n101-k10-C51-V5	542.0	542	**100.00**	1.5	542	**100.00**	57.8
M-n121-k7-C61-V4	707.7	719	**101.60**	2.2	720	101.74	98.3
M-n151-k12-C76-V6	629.9	659	**104.62**	3.2	659	**104.62**	113.1
M-n200-k16-C100-V8	744.9	791	**106.19**	5.3	805	108.07	158.7
G-n262-k25-C131-V12	2863.5	3249	**113.46**	6.2	3319	115.91	193.6
Average			**105.18**			*106.07*	

Table 3: Computational results on GVRP large instances with $n/p = 2$. Bold entries correspond to the best solution quality index for each row.

		ALNS			ITS		
Instance	LB	UB	100 × UB/LB	Time (sec.)	UB	100 × UB/LB	Time (sec.)
M-n101-k10-C34-V4	458.0	458	**100.00**	0.9	458	**100.00**	36.8
M-n121-k7-C41-V3	527.0	527	**100.00**	1.2	527	**100.00**	63.3
M-n151-k12-C51-V4	465.6	483	**103.74**	1.9	483	**103.74**	85.5
M-n200-k16-C67-V6	563.1	605	**107.44**	3.0	605	**107.44**	108.4
G-n262-k25-C88-V9	2102.4	2476	117.77	4.9	2463	**117.15**	134.3
Average			*105.79*			**105.67**	

Table 4: Computational results on GVRP large instances with $n/p = 3$. Bold entries correspond to the best solution quality index for each row.

	Number of	Number of		UTS			ITS	
Instance	customers	depots	UB	100 × UB/Best	Time (sec.)	UB	100 × UB/Best	Time (sec.)
pr01	48	4	1074.1	**100.00**	43	1077.7	100.33	54
pr02	96	4	1738.1	100.07	156	1737.0	**100.00**	245
pr03	144	4	2418.9	100.29	294	2411.9	**100.00**	499
pr04	192	4	2863.0	100.57	432	2846.8	**100.00**	761
pr05	240	4	3047.1	100.64	588	3027.6	**100.00**	917
pr06	288	4	3640.1	**100.00**	767	3698.8	101.61	1328
pr07	72	6	1413.6	**100.00**	93	1414.8	100.08	161
pr08	144	6	2088.7	**100.00**	307	2105.7	100.81	521
pr09	216	6	2727.2	**100.00**	530	2763.8	101.34	1085
pr10	288	6	3515.2	**100.00**	790	3520.4	100.15	1776
pr11	48	4	926.6	**100.00**	62	926.6	**100.00**	77
pr12	96	4	1457.7	101.46	220	1436.8	**100.00**	326
pr13	144	4	2060.7	101.58	375	2028.6	**100.00**	666
pr14	192	4	2251.6	**100.00**	580	2262.3	100.48	905
pr15	240	4	2498.0	**100.00**	674	2503.5	100.22	1186
pr16	288	4	2868.8	**100.00**	993	2936.9	102.38	1580
pr17	72	6	1171.2	**100.00**	127	1171.9	100.06	213
pr18	144	6	1826.6	**100.00**	356	1845.6	101.04	709
pr19	216	6	2294.0	100.08	720	2292.1	**100.00**	1593
pr20	288	6	3112.4	104.29	871	2984.3	**100.00**	1910
Average				100.45			**100.43**	

Table 5: Computational results on MDVRPTW instances. Bold entries correspond to the best solution quality index for each row.