Large composite neighborhoods for the capacitated location-routing problem

Working Paper DPO-2017-01 (version 2, 16.01.2017)

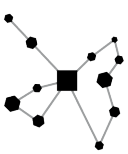
Michael Schneider and Maximilian Löffler
{schneider|loeffler}@dpo.rwth-aachen.de
Deutsche Post Chair – Optimization of Distribution Networks
RWTH Aachen University, Germany

Abstract

The capacitated LRP (CLRP) jointly takes decisions on the location of capacitated depots and the routing of capacitated vehicles to serve a set of customers with known demands from the opened depots. In city logistics, the CLRP is used for the planning of single-echelon distribution networks. In this application context, large-scale CLRP instances with a high number of potential depot locations (representing concrete candidates for running urban depots) and a high number of customers (due to densely populated urban areas) are particularly relevant. We introduce a tree-based search algorithm (TBSA) that explores the space of depot configurations in a tree-like fashion using a customized first improvement strategy. In the routing phase the multi-depot vehicle-routing problem defined by the depot configuration is solved with a granular tabu search that uses a large composite neighborhood described by 14 operators.

In numerical studies, we find that using the large composite neighborhood has a positive impact on the effectivity, efficiency and robustness of TBSA. We show that the tradeoff between run-time and solution quality of TBSA can be controlled by the number of total iterations and the strength of the sparsification in the granular search, and we investigate a fast, a basic, and a quality-oriented variant of TBSA. On the commonly used small to medium-sized CLRP benchmark sets from the literature, the fast version is able to dominate all previously published heuristic solution methods, i.e., it achieves better solution quality within shorter run-times. The basic and quality-oriented variant provide outstanding solution quality within reasonable run-times; on average, both variants achieve negative gaps to the previous best-known solutions. Thus, on the considered benchmark sets, the three variants together form the Pareto frontier of heuristic solution methods for the CLRP. Moreover, TBSA matches or improves the large majority of previous best-known solutions on these instances. Finally, TBSA is able to solve newly generated large-scale instances with up to 600 customers and 30 depots with reasonable run-times, and convincing scaling behavior and robustness. Additional experiments on open LRP benchmarks confirm the performance of TBSA.

Keywords: *location-routing problem, city logistics, large composite neighborhood, tree-based search, granular tabu search, large-scale instances*



Deutsche Post
Chair - Optimization of
Distribution Networks

RWTH AACHEN
UNIVERSITY

1 Introduction

Location-routing problems (LRPs) jointly take decisions on the location of depots and the routing of vehicles. It is well-known that making these decisions independently of one another may lead to strongly suboptimal planning results (Salhi and Rand 1989). Consequently, LRPs have been studied for decades (for earlier surveys, see Min et al. 1998, Nagy and Salhi 2007), and the research community has been very active in the last years (for several recent reviews, see Lopes et al. 2013, Drexl and Schneider 2014a,b, Prodhon and Prins 2014, Cuda et al. 2015, Albareda-Sambola 2015). Among their many fields of application, LRPs are of great importance to render decision support for the planning of single-echelon (Stenger et al. 2012) and multi-echelon (Cuda et al. 2015) distribution networks in city logistics. More precisely, LRPs assist in determining the locations of urban depots from which customers are served on vehicle routes. In this application context, LRPs often have the following characteristics:

- Potential depot locations are not representatives of promising regions or industrial areas (which is often the case when planning long-distance distribution networks) but represent concrete candidates for running a depot.
- Route planning for urban distribution often has to deal with a large number of customers due to densely populated urban areas.

These two characteristics lead to LRP instances with a large number of potential depot locations and a large number of customers to visit.

In this paper, we address the (single-echelon) capacitated LRP (CLRP), which can be defined as a graph theoretical problem as follows: We are given a set of depots $I = \{1, \dots, m\}$ and a set of customers $J = \{1, \dots, n\}$. With every depot $i \in I$, a capacity w_i and an opening cost o_i are associated. Every customer $j \in J$ has a positive demand d_j. At every depot, a fleet K with an unlimited number of identical vehicles with capacity Q and utilization cost F is available. Let $G = (V, A)$ be a directed graph with vertices $V = I \cup J$ and arcs $A = \{(v, w) \mid v \in I, w \in J\} \cup \{(v, w) \mid v, w \in J, v \neq w\} \cup \{(v, w) \mid v \in J, w \in I\}$. For each arc in A, a cost c_{vw} for traveling between two vertices v and w is given. The goal of the CLRP is to determine the depots to open and the routes from the depots to serve all customers, so that each customer is visited exactly once, each vehicle route ends at the depot from which it originated, and vehicle and depot capacities are respected. The objective function of the CLRP is to minimize total cost consisting of depot costs, vehicle costs and travel costs.

Due to its computational complexity and practical relevance, the CLRP has attracted a major effort of the research community to develop high-quality solution approaches. The most successful exact methods (Baldacci et al. 2011, Contardo et al. 2014b) are based on cut-and-column generation, and both methods rely on the decomposition of the problem into a set of multi-depot vehicle-routing problems (MDVRPs) and their exact solution. The two methods are able to solve the largest part of the benchmark instances from the literature with up to 100 customers and 10 depots, however, run-times are sharply rising for the larger of the instances. The method of Contardo et al. (2014b) is even able to solve 4 instances with 150/199 customers and 14 depots, but using an average run-time of 176 hours. This shortcoming of exact methods motivates the development of effective and efficient heuristic solution approaches for the CLRP. The most successful heuristic methods use a variety of paradigms: adaptive large neighborhood search (ALNS, see Hemmelmayr et al. 2012), simulated annealing (SA, see Yu et al. 2010), ant colony optimization (Ting and Chen 2013), variable neighborhood search (Pirkwieser and Raidl 2010, Escobar et al. 2014a), greedy randomized adaptive

search procedure (Duhamel et al. 2010, Contardo et al. 2014a) and matheuristic approaches integrating ILP techniques (Prins et al. 2007, Pirkwieser and Raidl 2010, Escobar et al. 2013, Contardo et al. 2014a). For summaries of the individual works, we refer the reader to the surveys of Lopes et al. (2013), Prodhon and Prins (2014), and Drexl and Schneider (2014b).

Apart from Alvim and Taillard (2013), who tackle extremely large instances of an LRP model without depot capacity constraints, we are not aware of methods specifically designed for large-scale problems. However, such problems do not only better reflect the characteristics of city logistics tasks but also of other practical applications, and it appears a worthwhile but challenging undertaking to develop a heuristic method that is able to obtain convincing solution quality within acceptable run-times. Moreover, a large-scale CLRP benchmark will help to provide a more meaningful comparison of metaheuristic approaches: All of the above listed heuristics achieve relatively similar solution quality on the commonly used small to medium-sized CLRP benchmarks of Tuzun and Burke (1999), Barreto (2004), and Prins et al. (2004). The methods providing the best solution quality have rather high run-times and therefore may be less suitable for solving large-scale instances.

We propose a new metaheuristic algorithm called tree-based search algorithm (TBSA) that consists of:

1. a location phase that explores the space of depot configurations in a tree-like fashion using a customized first improvement strategy. A tree-based search has already been successfully applied in the vehicle-routing context by Martinelli and Contardo (2015), who address a capacitated VRP (CVRP) with quadratic cost structure. Contrary to our approach, the authors always search the entire search tree and limit the branching degree and the search depth of the tree to fixed values.

2. a routing phase that uses a granular tabu search (GTS) to solve the MDVRP given by the depot configuration. The GTS uses a large composite neighborhood defined by 14 neighborhood operators, which is found to have a positive effect on the effectivity, efficiency and robustness of TBSA in the numerical studies. Contrary to many other works using granular searches, we define the reduced arc set by considering a number of shortest arcs for each depot and customer vertex instead of the shortest arcs in the entire arc set.

In the numerical studies, we show that the tradeoff between run-time and solution quality of TBSA can be adjusted by modifying the number of total iterations and the granularity features, resulting in a fast (TBSA$_{speed}$), basic (TBSA$_{basic}$), and quality-oriented (TBSA$_{qual}$) variant of TBSA. These variants achieve very convincing results on the commonly used benchmark instances from the literature. In fact, the fastest variant TBSA$_{speed}$ is able to dominate all previously published heuristic CLRP solution methods, i.e., it achieves better solution quality within shorter run-times than all other methods. The variants of TBSA putting more emphasis on solution quality, i.e., TBSA$_{basic}$ and TBSA$_{qual}$, on average show negative gaps to the previous best-known solutions (BKS) from the literature, thus providing outstanding solution quality within reasonable run-times. The three variants together form the Pareto frontier of heuristic CLRP methods on the benchmark instances from the literature. On the 79 instances contained in these benchmark sets, TBSA was able to match 51 and improve 27 of the previous BKS during the overall testing of our algorithm; the quality-oriented variant TBSA$_{qual}$ is able to match the previous BKS in 46 cases and to improve in 24 cases. Finally, we introduce a new CLRP benchmark set that contains a variety of large-scale instances with up to 600 customers and 30 potential depot locations. TBSA$_{basic}$ exhibits reasonable run-times, good scalability, and a robust solution behavior on these instances. Additional experiments on open LRP (OLRP) benchmarks confirm the convincing performance of TBSA.

Besides the above given references to high-quality metaheuristics and exact methods for the CLRP and to LRP survey articles, works on granular neighborhoods are relevant for this paper. Granular neighborhoods, originally proposed by Toth and Vigo (2003), rely on the definition of a local search move by means of a so-called generator arc, which uniquely identifies a move. The reduced set of generator arcs used to define a granular neighborhood is determined by applying a so-called sparsification method to the original arc set. Sparsification is often based on arc cost, and only the arcs whose costs are below a given threshold are considered as generator arcs. Granular neighborhoods have been successfully used to address the CVRP (Toth and Vigo 2003), the MDVRP either as standalone problem (Escobar et al. 2014b) or as routing part of the CLRP (Prins et al. 2007, Escobar et al. 2013, 2014a), and several other routing problems (see, e.g., Branchini et al. 2009, Labadie et al. 2012, Jin et al. 2012, Kirchler and Wolfler Calvo 2013, Berbotto et al. 2014).

Our large composite neighborhood is in some sense related to large neighborhoods in ALNS (see, e.g., Pisinger and Ropke 2010). However, contrary to ALNS, which typically uses different operators to diversify and intensify the search, our large composite neighborhood solely aims at a strong intensification effect.

The remainder of this paper is structured as follows. We give a detailed description of the components of TBSA in Section 2. Section 3 presents extensive numerical studies to assess the performance of TBSA, the effect of its algorithmic components and its scaling behavior on large-scale CLRP instances. A short conclusion is given in Section 4.

2 Tree-based search algorithm for the CLRP

The overall structure of TBSA is presented in Figure 1 and its main components are detailed in this section.

To generate an initial solution, all customers are assigned to their closest depot neglecting depot capacities, and routes for each depot are generated in cheapest insertion fashion. Subsequently, the routes are improved by means of a local descent using a large composite neighborhood of 14 different types of Relocate, Exchange, and Split operators as described in Section 2.2. Then, TBSA iterates between a location phase (Section 2.1) and a routing phase (Section 2.2). In the location phase, a neighborhood search on depot configurations is carried out in tree-like fashion using a customized first improvement strategy, which carries out a transition to the neighboring configuration only if the objective value of this configuration improves the overall best solution. The selected neighbor from the depot configuration neighborhood is used as target depot configuration and either a solution possessing the target configuration is fetched from a cache of complete CLRP solutions (see Section 2.1), or a transformation heuristic translates the current solution into a solution with the target depot configuration (Section 2.3). A routing phase is called to evaluate the resulting solution. Here, a MDVRP with capacitated depots and an unlimited number of available vehicles for each depot, that results from the current depot configuration, is solved using a GTS.

TBSA runs through a number of different intensity levels $\ell \in \{1, \ldots, \ell_{max}\}$, starting at level 1. When entering a higher intensity level, the number of depot configurations to be evaluated is reduced by discarding non-promising configurations based on their performance in the previous intensity level. More precisely, the worst λ_w percent of all configurations that were evaluated with a routing phase in the previous intensity level are marked as forbidden. Moreover, the number of iterations of the location phase is decreased, and at the same time, the thoroughness of the routing phase is enhanced by increasing the number of iterations. An intensity level transition from the current intensity level ℓ takes place if the best overall solution S^* has not

$S \leftarrow$ initial solution
$S^* \leftarrow S$
add S^* to $listOfBestSolutions$
$\ell \leftarrow 1$ {set intensity level}
repeat
  $S \leftarrow locationPhase(S)$ {calls routing phase to evaluate solutions}
  update $listOfBestSolutions$
  if $f(S) < f(S^*)$ **then**
    $S^* \leftarrow S$
  end if
  if all configurations in neighborhood graph visited or S^* not improved for η_ℓ^{loc} iterations **then**
    $\ell \leftarrow \ell + 1$
    $S \leftarrow S^*$
  end if
until $\ell = \ell_{max} + 1$
$S^* \leftarrow finalizationPhase(listOfBestSolutions)$
return S^*

Figure 1: Pseudocode of TBSA

improved for η_ℓ^{loc} iterations of the location phase. The location phase of the next intensity level begins with the overall best solution found so far.

After TBSA has run through all intensity levels, a finalization phase is carried out (see Section 2.4). This phase consists of a number of finalization levels that are similar to the intensity levels: the most promising depot configurations are more thoroughly evaluated by carrying out routing phases with more iterations and stronger intensification.

2.1 Location phase

The location phase carries out a tree-like neighborhood search on depot configurations. Let $C(S_{curr})$ denote the depot configuration of the current solution S_{curr} from which the search starts. The depot configuration neighborhood $\mathcal{N}_{depot}(C(S_{curr}))$ is defined by the operators *Swap* and *Invert*. Swap closes one depot and opens another one instead; Invert closes an open depot or opens a closed one.

First, infeasible configurations in the depot configuration neighborhood whose cumulative depot capacities are not sufficient to serve the total customer demand are marked as forbidden and will not be evaluated again through the course of the algorithm. In addition, we check whether the lower bound for the objective value of a configuration in the neighborhood is higher than the objective value of the overall best found solution. All configurations for which this is true are also marked as forbidden. As lower bound, we use the sum of the costs of the open depots in the configuration, the vehicles necessary for satisfying the cumulative demand of the customers, and the travel costs of the minimum spanning forest.

In order to evaluate a yet unvisited configuration in the neighborhood, a routing phase is called (see Section 2.2 for details on the routing phase). We search the configuration neighborhood of the current solution using a customized first improvement strategy: As soon as we have found a configuration whose objective value determined by the routing phase improves upon that of the best overall solution, the corresponding CLRP solution becomes the new current solution and the remaining configurations in the neighborhood stay unvisited for the moment, i.e., they are not evaluated with a routing phase .

Before the configuration neighborhood $\mathcal{N}_{depot}(C(S_{curr}))$ of the current solution is searched, all contained configurations are ordered in a hierarchical fashion that gives higher priority to more promising solu-

tions. This order determines the sequence in which the neighborhood is later traversed, and it is defined as follows:

- The first criterion is the intensity level at which the configuration has last been evaluated with a routing phase: configurations evaluated with a routing phase at higher intensity levels have a higher priority, configurations that have never been evaluated with a routing phase come last.
- The configurations that were evaluated on the same intensity level are ordered either according to their tentative objective function value determined by the routing phase or according to the following rating function (in case they have never been evaluated with a routing phase).
- The rating function R estimates the spatial balance of the capacity supply given by a depot configuration C and the customer demand of the given problem instance. Let $\bar{c}_{ij} = c_{ij} + c_{ji}$, then R is defined as:

$$\mathrm{R}(C) = \sum_{i \in C} \left(\left| \sum_{j \in J} \left(d_j \cdot 0.95^{\bar{c}_{ij}} \right) - \sum_{l \in C} \left(w_l \cdot 0.95^{\bar{c}_{li}} \right) \right| + o_i \right). \tag{1}$$

  The first part of the expression within the absolute value signs measures the density of demands, the second part the supply density at the depots of C. Thus, the absolute value measures the balance of supply and demand in the given configuration, with perfect balance resulting in a value of zero for the entire expression. Finally, the costs of the depots are also taken into account. The smaller the value of the rating function, the better the balance of supply and demand and/or the cheaper the opening costs of depots. Therefore, configurations with smaller values are preferred.

The thus defined traversal order of the configuration neighborhood has an intensification effect because it puts more emphasis on promising configurations that have achieved good objective values in previous intensity levels. Due to the large number of depot configurations to investigate for larger problem instances, we forgo strong diversification in the location phase. Note that after determining the order, all configurations contained in the neighborhood have either been evaluated by a previous routing phase or have been assessed by the above rating function, i.e., a tentative objective function value is known for each of the configurations.

Evaluation of a move consists of two steps:

1. The neighbor configuration is used as so-called target configuration. In case a CLRP solution possessing the target configuration is available from the cache, we directly go to step 2. Otherwise, we transform the current solution S_{curr} into a solution with the depot configuration C_{target}. This is achieved by a transformation heuristic T, whose details are described in Section 2.3 after the neighborhood operators of the routing phase have been introduced. In special cases, the transformed solution does not use all depots that are open in the target configuration, and thus, the depot configuration of the transformed solution is a subset of the target configuration, namely $C(T(S_{curr}, C_{target})) \subseteq C_{target}$.
2. Next, a routing phase is carried out starting from the transformed solution. It may open depots of the target configuration C_{target} that have not been opened by the transformation heuristic T, or it may close depots that have been opened by T.

Configurations evaluated with a routing phase are stored in the list $\mathcal{L}$ ordered according to their objective value given by the actual depot configuration and the routes determined in the routing phase. Both, the

actual depot configuration $C\left(T(S_{curr}, C_{target})\right)$ and the target depot configuration C_{target}, are associated with the objective value and inserted into the list. In case these configurations are already present in the list, their tentative objective function value is updated if it improves the already stored value, otherwise nothing happens. In addition, both configurations are marked as visited and will not be visited again in the current intensity level. If the search enters the next intensity level, all configurations are marked as unvisited. As mentioned earlier, for the first 50 entries in $\mathcal{L}$, we additionally maintain a cache of complete CLRP solutions, i.e., we also save the vehicle routes. Storing complete solutions for all visited depot configurations is not possible due to memory requirements.

As described above, the first neighbor improving the best overall solution becomes the new current solution. This means that the neighborhood is sometimes not completely explored on the first pass. If the neighborhood of the current solution is searched completely without finding an improving solution, the current configuration is reset to the best configuration C' from $\mathcal{L}$ which has unvisited neighbors. Then, the search continues with the best unvisited configuration C'' in the neighborhood $\mathcal{N}_{depot}(C')$. In case we have stored a complete CLRP solution for C'' in a previous intensity level, this solution is used as starting point for the routing phase. Otherwise, the transformation heuristic T is applied to S_{curr} with C'' as target configuration in order to define the starting point of the routing phase. If the list of configurations with unexplored neighborhoods is empty, i.e., there are no unvisited neighbor configurations in this connected component of the neighborhood graph, the search continues with a randomly selected unvisited configuration. This can happen if the neighborhood graph is not connected because of forbidden configurations. As shown in the pseudocode in Figure 1, the next intensity level starts when either the entire neighborhood graph is explored or the best solution has not improved for η_ℓ^{loc} iterations.

2.2 Routing phase

The location phase calls the routing phase in order to evaluate depot configurations. In the routing phase, we use a GTS to solve the MDVRP defined by the given depot configuration. The initial solution of the routing phase is either determined by the transformation heuristic described in Section 2.3, or the routing phase starts from a routing solution that has been determined in a previous intensity level.

During the search process, the GTS is allowed to visit solutions that are infeasible concerning depot capacities and vehicle capacities. The solutions are penalized by means of a generalized cost function as described in Section 2.2.1. GTS uses a composite neighborhood defined by 14 operators detailed in Section 2.2.2. In order to speed up the search, granular neighborhoods are used (Section 2.2.3). As tabu criterion, we forbid the reinsertion of an arc that has been removed by the GTS for a tenure of τ iterations, where τ is randomly drawn from a uniform distribution on the interval $[\tau_{min}, \tau_{max}]$. In each iteration, the best non-tabu move is carried out. A move which is tabu but improves on the best solution found during the routing phase is always permitted. To diversify the search, we use a continuous diversification mechanism that biases the search towards solutions with arcs that have seldom been included in previous iterations (Section 2.2.4).

The routing phase first runs until a feasible routing solution is found, then the search continues until the best found solution has not improved for η^{gts} iterations. We define the total number of iterations in this manner because it is very important for the quality of TBSA to find feasible routing solutions for given depot configurations in order to allow for a meaningful comparison of the objective function values of different configurations.

2.2.1 Generalized cost function

We use a generalized cost function that penalizes violations of depot and vehicle capacities. Let $P_{dCap}(S)$ be the cumulative capacity violation at the depots for a given solution S and $P_{vCap}(S)$ the cumulative capacity violation of the vehicles. Then, the generalized cost function $f_{gen}(S)$ is:

$$f_{gen}(S) = f(S) + \rho(S) \cdot \big(\alpha P_{dCap}(S) + \beta P_{vCap}(S)\big),$$

where $f(S)$ denotes the objective function value of S, α and β are penalty factors, and $\rho(S)$ is a normalization factor.

The penalty factors α and β are initialized to $\alpha_0 = \beta_0$ and then dynamically updated during the search in order to guide the search towards feasible solutions if a large number of recently visited solutions is infeasible and in order to allow for more diversification in case most visited solutions are feasible. To this end, the respective penalty factor is multiplied with δ if the depot or vehicle capacity is violated in the current solution and divided by δ in case the solution is feasible in terms of depot or vehicle capacity.

The penalty factors are not strictly bounded to an interval as commonly done in the literature. Instead, we allow the penalty factors to increase and decrease arbitrarily strongly. The first is important to ensure that GTS finds feasible solutions quickly, the second helps to strongly diversify the search. With unbounded penalty factors, a large number of feasible (infeasible) solutions is needed to reduce (increase) the penalty factor to an adequate value after a longer series of infeasible (feasible) solutions. We counteract this negative effective as follows: if one of the penalty factor has risen beyond a given upper bound (fallen below a lower bound) after a series of infeasible (feasible) solutions, the factor is set to this upper (lower) bound as soon as the first feasible (infeasible) solution is found. More precisely, if S exhibited capacity violations on depots and/or vehicles in the previous iteration, we set $\alpha = \max(\alpha_{min}, \alpha\delta)$ and/or $\beta = \max(\beta_{min}, \beta\delta)$. Else, we set $\alpha = \min(\alpha_{max}, \alpha/\delta)$ and/or $\beta = \min(\beta_{max}, \beta/\delta)$.

Although the penalty factors are not bounded, they are reset according to the above rules in order to render the penalty mechanism more reactive. To make it possible that the algorithm visits infeasible solutions on instances where the demand and capacities (and thus the penalties) have a higher order of magnitude compared to the objective value and, vice versa, that the algorithm is guided to feasible solutions in case demand and capacities have a lower order of magnitude, the normalization factor $\rho(S)$ is used to relate the magnitude of the objective function value and the capacity penalties. In other words, we use $\rho(S)$ to decouple the value of the penalty term from the specific problem instance at hand. In the design of the normalization factor, we assume a uniform distribution of the overall demand over the customers, in which case the average capacity violations occurring during the search lie within the same order of magnitude as the normalization factor ρ defined as follows:

$$\rho(S) = \frac{|J| \cdot f(S)}{\sum_{j \in J} d_j}.$$

2.2.2 Large composite neighborhood

In each iteration, the GTS carries out the best non-tabu move in a large composite neighborhood defined by 14 operators: the well-known 2-Opt*, 5 types of Split, 4 types of Relocate and 4 types of Exchange operators. The purpose of this extended operator set is intensification, i.e., to achieve a high convergence rate and thus obtain very good routing solutions in short time. The numerical studies in Section 3 show that the large composite neighborhood has a positive effect on the performance and robustness of TBSA.

We define all operators using the generator arc principle introduced in Section 1. Let (v, w) denote the generator arc, v_- and v_+ the predecessor and successor of vertex v, v_f and v_l the first and the last customer in the route which contains v, and v_d the depot to which the route belongs. With this notation, the operators are defined as depicted in Figure 2:

- 2-Opt*, is used in inter-route and intra-depot fashion, i.e., it is not evaluated for routes starting at different depots. It removes one arc from two routes and reconnects the first part of the first route with the second part of the second route and vice versa (see Figure 2(a)). The 2-Opt* operator is able to close a route if v_+ and w_- are depot vertices.

- Split operators remove x and insert $x + 1$ arcs ($x \in \{1, 2\}$) in order to move parts of the existing route to a new route located at the same or a different depot. Five types of Split operators are used in the search process, see Figures 2(b)–2(f). Split-0 splits a route into two new routes and is only defined in intra-depot fashion. Split-1 creates a new route containing a single customer and Split-2 moves two consecutive customers to a newly created route. These two operators are used in intra- and inter-depot fashion. Split-b creates a new route with all predecessors of vertex v_+, while Split-f moves all successors of w_- to a new route. These operators are only used in inter-depot fashion since they are equivalent to the Split-0 operator in the intra-depot case. Split operators are able to raise the number of vehicles used in the solution by one.

- Relocate operators move customers between routes or within a route, see Figures 2(g)–2(j). All Relocate operators are defined in intra- and inter-route as well as intra- and inter-depot fashion. Relocate-1 moves one customer, Relocate-2 two consecutive customers. Relocate-b and Relocate-f relocate all predecessors of v_+ or successors of w_- respectively. The Relocate operators are able to reduce the number of vehicles used in the solution by one.

- Exchange operators swap customers between routes or within a route, see Figures 2(k)–2(n). They are defined in analogous fashion to the Relocate operators. Exchange-1 and Exchange-2 are used in intra- and inter-route and intra- and inter-depot fashion. Exchange-b and Exchange-f are defined as inter-route and inter-depot moves since they are equivalent to the 2-Opt* operator in the intra-depot case.

For all presented operators, the change in objective function value, route capacity violation, and depot capacity violation can be computed in constant time.

2.2.3 Granular search

To speed up the search, we use granular neighborhoods for the 2-Opt* and all types of Relocate and Exchange operators. Only for the Split operators, all generator arcs—which in this case corresponds to all depot arcs—are used and no sparsification takes place. Pretests have shown that keeping all arcs has a strong positive effect on solution quality but only a slight negative effect on the run-time of the algorithm. The latter is due to the fact that only $\mathcal{O}(|I||J|)$ generator arcs exist for the Split operators.

The granular neighborhoods are defined by a reduced set of generator arcs A_κ given by (i) for each customer, the κ percent of the shortest arcs to other customers, (ii) for each customer, the κ percent of the shortest depot arcs, i.e., arcs from the depot to the customer or vice versa, and (iii) the arcs of the current best found solution of the routing phase.

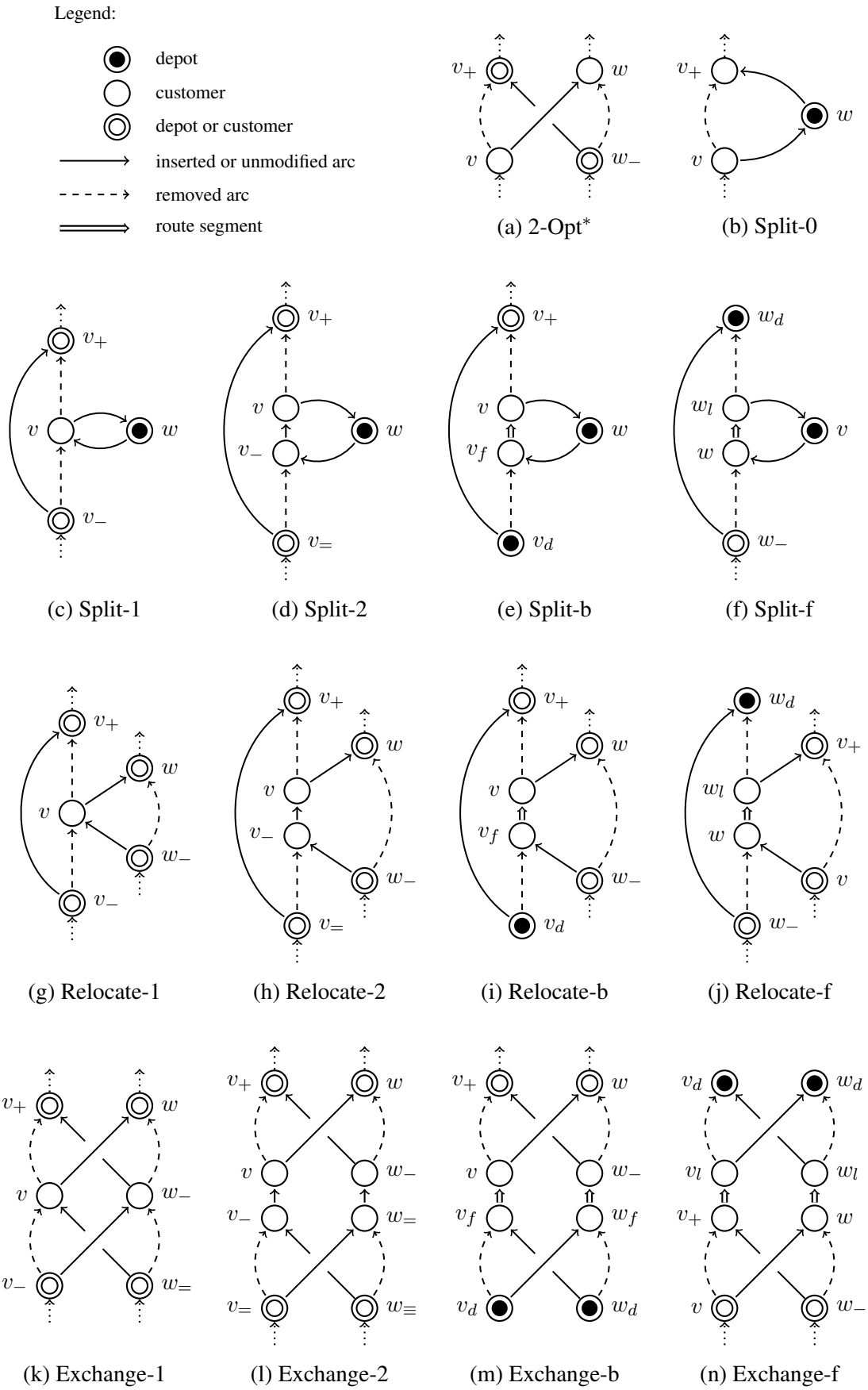


Figure 2: Neighborhood operators of the GTS

We dynamically adapt the strength of the sparsification by updating the granularity threshold κ after every η_κ iterations. We compare the best found objective value of the last η_κ iterations with the best value of the previous η_κ iterations. In case the objective value has improved, κ is divided by δ_κ, in case of a deterioration, κ is multiplied by δ_κ. The granularity threshold κ is restricted to the interval $[\kappa_{min}, \kappa_{max}]$.

2.2.4 Continuous diversification

As sole diversification technique, GTS uses continuous diversification, originally proposed by Cordeau et al. (1997). The general idea is to add a penalty term to the cost of a solution that depends on the similarity of the solution to previously visited solutions. In this way, the search is guided towards unexplored regions of the solution space. We keep an insertion count $\Lambda(a)$ for every arc $a \in A$. Every time a move is carried out that inserts the set of arcs A_{move}, $\Lambda(a)$ is incremented by one for every arc $a \in A_{move}$ that is inserted by the move. Moves deteriorating the generalized cost value, i.e, moves from S to S' with $f_{gen}(S') \geq f_{gen}(S)$ are evaluated with the following extended generalized cost function:

$$f^{ext}_{gen}(S') = f_{gen}(S') + \sigma \cdot f(S') \cdot \sum_{a \in A_{move}} \Lambda(a).$$

The second term describes the diversification penalty with diversification intensity σ, which is set to high values during the location phases and low values during the finalization phase (see Section 3.2). The insertion count of all arcs is reset to zero as soon as a new overall best solution is found.

2.3 Transformation heuristic

The transformation heuristic T is used in the location phase as part of the move evaluation. Here, the selected neighbor from the depot configuration neighborhood is used as target depot configuration C_{target}, and T translates the current solution S_{curr} into a solution with this target depot configuration. The goal of the transformation heuristic is twofold:

1. For all newly opened depots, i.e., for all depots in $C_{target} \backslash C(S_{curr})$, we want to avoid that the opening decision is quickly reversed by the search because too few customers are assigned to the newly opened depots. To achieve this, we assign to these depots all the customers for which the respective depot is the nearest.

2. We want to keep as much as possible of the routing of the current solution because this routing solution was previously determined by a routing phase and is already of good quality. To achieve this, we try to conserve as much as possible the routes starting at depots that are open in the current solution and in the target configuration, and the routes starting at depots that are only open in the current solution but not in the target configuration.

In the first step of the transformation heuristic, we pretend that all depots in $C_{target} \cup C(S_{curr})$ are available and determine the nearest depot for each customer. Then, for each depot i in $C_{target} \setminus C(S_{curr})$, we extract the customers for which i is the closest depot from their current routes and insert them into routes starting from i in cheapest insertion manner using the neighborhood operators Split-1 and Relocate-1. In this step, we allow infeasible solutions with respect to depot capacities, and we ignore the violations, i.e., we set $\alpha = 0$. However, we aim at generating capacity-feasible vehicle routes and therefore do not allow violations of vehicle capacities. In case there are no customers for which i is the nearest depot, T does not open i.

Second, we keep the routes of all depots in $C_{target} \cap C(S_{curr})$. As explained above, these routes are only modified by removing those customers for which the nearest depot is in $C_{target} \setminus C(S_{curr})$.

Third, all routes of S_{curr} that start at a depot that is not open in C_{target} are cut off from their depot by connecting the last customer on the route with the first customer. The resulting subtours are reconnected in a cheapest insertion fashion by means of the operators 2-Opt*, Split-0, Split-1, Split-2, Relocate-1, and Relocate-2. As generator arcs, we use all arcs (v, w) where vertex v is part of a subtour and w is either a depot in C_{target} or a customer on a route that is connected to a depot in C_{target}. In this step, infeasible solutions with regard to depot capacity are also forbidden.

Finally, a local descent using the Relocate-1 operator is performed on the resulting solution. Here, again neither depot capacity nor vehicle capacity violations are allowed. In this way, we guide T towards feasible solutions.

2.4 Finalization phase

After TBSA has completed all intensity levels, the finalization phase tries to improve the routing part of the most promising solutions. In three finalization levels, we evaluate an ever smaller number of high-quality solutions with increasing thoroughness and intensification by raising the total number of iterations and reducing the diversification intensity of the GTS.

3 Numerical studies

This section starts with an introduction of the benchmark instances used to evaluate TBSA (Section 3.1) and a description of the parameter setting of the basic variant of TBSA, called TBSA$_{basic}$ in the following (Section 3.2). Section 3.3 introduces three additional variants of TBSA that are used (i) to demonstrate that the tradeoff between solution quality and run-time can successfully be controlled with TBSA, and (ii) to investigate the effect of the large composite neighborhood on the quality of TBSA. The respective studies are presented in Section 3.4. In the same section, we also compare the performance of TBSA to the state-of-the-art CLRP methods on the classical benchmarks from the literature. Moreover, we address these benchmarks as OLRP instances and compare our results to the OLRP method proposed by Yu and Lin (2015). Finally, Section 3.5 investigates the ability of TBSA to solve our newly proposed CLRP benchmark set, with a special focus on large-scale instances.

3.1 Benchmark instances

The computational experiments with TBSA are performed on two different types of benchmarks: (i) classical CLRP benchmarks consisting of a total of 79 problem instances (which can also be addressed as OLRP benchmarks), and (ii) a set of 202 self-generated instances, including large-scale instances with up to 600 customers and 30 depots.

The first classical benchmark set, introduced by Tuzun and Burke (1999) and called Tuzun-Burke set, consists of 36 instances with $n \in \{100, 150, 200\}$ customers, $m \in \{10, 20\}$ uncapacitated depots, and capacitated vehicles. The set contains instances with varying density in customer distribution and clusters with different sizes. Travel costs are given by the Euclidean distance as floating point number. The second classical benchmark, called Prodhon set, was proposed by Prins et al. (2004) and consists of 30 instances with $n \in \{20, 50, 100, 200\}$ customers and $m \in \{5, 10\}$ depots. All instances are restricted in terms of vehicle and depot capacity. The instances in this benchmark are named n–m–tc, where t denotes the instance type

and c indicates the vehicle capacity: In type 1 instances, the customers are evenly distributed in the plane. In type 2 (3) instances, customers are clustered into 2 (3) clusters of equal size. Instances with $c = a$ have a vehicle capacity of 70, instances with $c = b$ have a vehicle capacity of 150. Travel costs for these instances are given by the Euclidean distance, multiplied by a factor of 100 and rounded up to the next integer. The third classical benchmark, proposed by Barreto (2004) and called Barreto set, contains 13 instances of which most are derived from classical CVRP instances. The instances feature up to 150 customers and 10 depots, vehicle capacities are generally limited, and in most of the instances depot capacities are also restricted. Travel costs are again given by the Euclidean distance as floating point number.

In addition, we generate a set of 202 new instances that follow the structure of the Prodhon instances for some part but also contain new ideas on how to generate hard-to-solve instances and additionally comprise large-scale instances. The set contains instances with $n \in \{100, 200, 300, 400, 500, 600\}$ customers and $m \in \{5, 10, 15, 20, 25, 30\}$ depots. More precisely, the following 11 groups defined by (n, m) pairs are considered: $(100, 5)$, $(100, 10)$, $(200, 10)$, $(200, 15)$, $(300, 15)$, $(300, 20)$, $(400, 20)$, $(400, 25)$, $(500, 25)$, $(500, 30)$, $(600, 30)$. For each group, four types of instances are created: Types 1, 2, and 3 are similar to those of the existing Prodhon instances. In type 4 instances, a quarter of the customers is evenly distributed in the plane, and the remaining three quarters of customers are clustered into three clusters of high density and equal size. Such settings in which uniformly distributed customers as well as customer clusters occur are present in the Tuzun-Burke set, but only without depot capacities. Instances with these characteristics are of practical relevance in both urban and in long-distance distribution. Moreover, from a computational perspective, this type is challenging for granular solution methods because short arcs (to connect customers in clusters) as well as longer arcs (to connect clusters to randomly distributed customers or randomly distributed customers to each other) are important for finding high-quality solutions. Therefore, we decided to include such instances.

For each of these four instance types, one instance for the following five subtypes 'a', 'b', 'c', 'd', and 'e' is generated:

'a': The instance parameters in this subtype are set to values resembling those of the original Prodhon instances. For instance sizes larger than the original ones, depot costs and capacities are extrapolated. The vehicle capacity is set to 70.

'b': Same as 'a', but the vehicle capacity is set to 150.

'c': Same as 'a', but the depot costs are divided by 1000. When depot costs are very small and are dominated by the routing costs, it is no longer possible for algorithms to enumerate the interesting depot configurations as is the case with high depot costs.

'd': These instances are constructed such that the minimum number of open depots to satisfy the total customer demand is approximately $I/2$. This is achieved by setting the depot capacities to double the overall demand divided by the depot count and multiplying each of them by a random factor uniformly distributed on the interval $[1, 1.02]$ and then rounding down to the next integer value.

'e': Compared to the other subtypes, these instances show larger differences in the depot capacities and the depot costs. The depots with higher capacity have higher costs and vice versa. There are more depots with small capacities than depots with high capacities. In this way, a large number of interesting configurations with both a lower number of high-capacity depots and a higher number of low-capacity depots exists.

For all instances, the depots are distributed evenly in the plane. Like in the original Prodhon instances, travel costs are given by the Euclidean distance, multiplied by a factor of 100 and rounded up to the next integer. In this way, we generate $11 \times 4 \times 5 = 220$ instances. However, for the (n, m)-groups $(100, 5)$, $(100, 10)$, and $(200, 10)$, instances of types 1, 2, and 3 and subtypes 'a' and 'b' are already available in the Prodhon set, and thus we end up with $220 - (3 \times 3 \times 2) = 202$ instances. The instances are available as online supplement.

3.2 Parameter setting

The parameters of TBSA$_{basic}$ have been set to adequate values during the development of the algorithm. The relatively high number of parameters is mostly related to the iteration numbers in the different intensity levels, the penalty mechanism, and the dynamic sparsification used by the algorithm. All these components have a significant positive effect on the performance of TBSA (and other routing and location-routing algorithms in the literature) but have the drawback of increasing the number of algorithm parameters. However, pretests showed that TBSA$_{basic}$ is very robust with regards to changes in the parameter values as long as the values stay within the same orders of magnitude as the values given below, which somewhat mitigates the drawback. Due to this observation, we forego an extensive parameter tuning and thus also avoid to overfit the parameters to the instances under consideration.

TBSA$_{basic}$ runs through $\ell_{max} = 3$ intensity levels. In level 1, a maximum of $\eta_1^{loc} = 750$ configurations without improvement are evaluated, and in the routing phase, GTS runs for $\eta_1^{gts} = 25$ iterations without improvement. For level 2, we set $\eta_2^{loc} = 200$ and $\eta_2^{gts} = 50$, and for level 3, we set $\eta_3^{loc} = 50$ and $\eta_3^{gts} = 250$. In an intensity level transition, the worst $\lambda_w = 80\%$ of the visited configurations are marked as forbidden.

On the first finalization level, the best $\vartheta_{fin_1} = 5$ solutions found during the intensity levels are reevaluated using a routing phase with $\eta_{fin_1}^{gts} = 1000$ stop iterations and $\sigma_{fin_1} = 2.0$. On the second finalization level, we improve the best $\vartheta_{fin_2} = 3$ solutions from the previous level with $\eta_{fin_2}^{gts} = 10000$ stop iterations and $\sigma_{fin_2} = 0.02$. Finally, the best solution ($\vartheta_{fin_3} = 1$) is evaluated by a routing phase with $\eta_{fin_3}^{gts} = 50000$ stop iterations and $\sigma_{fin_3} = 0.00002$.

Apart from the numbers of executed iterations, the parameters of GTS are equal for all intensity levels: The diversification intensity of GTS is set to $\sigma = 20$. The tabu tenure is drawn from the interval $[\tau_{min} = 6, \tau_{max} = 10]$. The penalty factors are set to $\alpha_0 = \beta_0 = 1$, $\alpha_{min} = \beta_{min} = 0.01$, and $\alpha_{max} = \beta_{max} = 100$. The factor δ is chosen such that the penalty rises from α_{min}/β_{min} to α_{max}/β_{max} after 100 infeasible iterations (respectively falls from α_{max}/β_{max} to α_{min}/β_{min} after 100 feasible iterations), i.e., $\delta = \exp(\log(100/0.01)/100) \approx 1.1$. The parameters to define the granular neighborhood are set to $\kappa_{min} = 0.05, \kappa_{max} = 0.2, \eta_\kappa = 50$, and $\delta_\kappa = \exp(\log(0.2/0.05)/5) \approx 1.32$ (analogous to δ but with 5 steps in the interval $[0.05, 0.2]$).

The described parameter setting is used by TBSA$_{basic}$, the TBSA variants presented in the next section partly change a small subset of the parameters. All variants of TBSA are implemented in C++ and the experiments are run on a dedicated node of a computing cluster with a Xeon E5 2670 processor at 2.60 GHz, 32 GB of RAM, and running SUSE Linux Enterprise Server 11. The RAM available to the process is restricted to 128 MB, and only a single core of the processor is used.

3.3 Variants of TBSA

To show that the tradeoff between solution quality and run-time achieved by TBSA can be successfully tuned by adjusting the number of iterations and the granularity settings of TBSA, we consider two additional variants of TBSA (besides TBSA$_{basic}$) in the following numerical experiments:

TBSA$_{speed}$ puts more emphasis on short run-times and less emphasis on solution quality. The parameter setting for TBSA$_{speed}$ is equal to that of TBSA$_{basic}$ except for (i) the numbers of iterations, which are $\eta_1^{loc} = 250$, $\eta_2^{loc} = 100$, $\eta_3^{loc} = 20$, $\eta_1^{gts} = 10$, $\eta_2^{gts} = 25$, and $\eta_3^{gts} = 100$, and (ii) the parameters of the finalization phase, which are $\kappa_{min} = \kappa_{max} = 0.05$, $\eta_{fin_1}^{gts} = 500$, $\eta_{fin_2}^{gts} = 5000$, and $\eta_{fin_3}^{gts} = 10000$.

TBSA$_{qual}$ puts more emphasis on solution quality and less emphasis on run-time. The parameters of TBSA$_{qual}$ are equal to those of TBSA$_{basic}$ except for the finalization phase. Here, we set $\kappa_{min} = 0.05$, $\kappa_{max} = 0.3$, $\eta_{fin_1}^{gts} = 5000$, $\eta_{fin_2}^{gts} = 50000$, and $\eta_{fin_3}^{gts} = 250000$. Increasing the numbers of iterations of the location phase compared to TBSA$_{basic}$ showed no significant improvement of solution quality in preliminary studies and was therefore not implemented.

Finally, we consider a third variant of TBSA to investigate the effect of our large composite neighborhood on the performance of TBSA:

TBSA$_{classic}$ restricts the composite neighborhood to the classical operators 2-Opt*, Split-1, Relocate-1, and Exchange-1. Apart from this, TBSA$_{classic}$ uses the same parameter setting as TBSA$_{basic}$.

3.4 Performance analysis of TBSA on the classical CLRP benchmark sets

The aim of this section is twofold. First, we compare the variants of TBSA, i.e., we investigate the effect of the large composite neighborhood on the effectivity, efficiency, and robustness of TBSA, and we study the possibility of tuning the tradeoff between solution quality and run-time by means of the number of iterations and the granularity setting (Section 3.4.1). Second, we compare the solution quality of the variants TBSA$_{speed}$, TBSA$_{basic}$, and TBSA$_{qual}$ to the state-of-the-art CLRP and OLRP algorithms from the literature (Section 3.4.2).

To this end, Table 1 presents an aggregate view on the performance of all relevant heuristic methods on the Tuzun-Burke, Prodhon, and Barreto benchmark: The upper part of the table shows the results of all TBSA variants (TBSA$_{classic}$, TBSA$_{speed}$, TBSA$_{basic}$, and TBSA$_{qual}$), and the best results that were obtained with any of the variants during the overall testing of our algorithm ($\overline{\text{TBSA}}$); the lower part of the table presents the results of the most successful CLRP heuristics in the literature. For each method, the average gap to the BKS (the BKS is taken from Contardo et al. (2014b) and Escobar et al. (2014a)) and the average run-time is reported for each separate instance set and as average over all three instance sets. The solution quality based on the best of several runs is reported as Δ_f^b, based on the average of several runs as Δ_f^a, and based on a single run as Δ_f. In addition, we translate the run-times of all methods into a common time measure based on the Passmark score (see www.passmark.com) of the processor used in the respective paper and the Xeon E5 2670 processor on which our numerical studies were run. Note that the score refers to the single-core performance of the processors because the numerical experiments in all papers were carried out on a single core. The run-times of methods that base their reported solution quality on the best of several runs are multiplied by the number of runs. We are aware that an entirely precise comparison of run-times is not possible due to different operating systems and programming languages, however, this is the best we can do to provide a fair comparison.

Benchmark	**TBSA**$_{classic}$			**TBSA**$_{speed}$			**TBSA**$_{basic}$			**TBSA**$_{qual}$			$\overline{\textbf{TBSA}}$
	$\Delta_f^b(\%)$	$\Delta_f^a(\%)$	$t(s)$	$\Delta_f^b(\%)$	$\Delta_f^a(\%)$	$t(s)$	$\Delta_f^b(\%)$	$\Delta_f^a(\%)$	$t(s)$	$\Delta_f^b(\%)$	$\Delta_f^a(\%)$	$t(s)$	$\Delta_f^b(\%)$
Tuzun-Burke	0.15	0.42	134.41	0.16	0.41	65.75	-0.08	0.12	324.84	-0.11	0.00	1004.65	-0.15
Prodhon	0.02	0.18	38.27	0.03	0.13	32.07	-0.06	-0.01	130.91	-0.07	-0.05	625.84	-0.08
Barreto	0.03	0.26	18.36	-0.01	0.07	10.17	-0.01	0.05	38.63	-0.01	0.04	171.43	-0.01
Average	**0.08**	**0.30**	78.80	**0.08**	**0.25**	43.82	**-0.06**	**0.06**	204.10	**-0.08**	**-0.01**	723.68	**-0.10**
Processor type	Xeon E5 2670												
Processor speed	2.60 GHz												
Passmark score	1652												
Corrected time	**78.80** $\times$ **5**			**43.82** $\times$ **5**			**204.10** $\times$ **5**			**723.68** $\times$ **5**			–

Benchmark	**DLPP**		**YLLT**		**HCC–500K**			**HCC-5000K**			**CCG**			**ELT**		**TC**		**ELBT**	
	$\Delta_f^b(\%)$	$t(s)$	$\Delta_f(\%)$	$t(s)$	$\Delta_f^b(\%)$	$\Delta_f^a(\%)$	$t(s)$	$\Delta_f^b(\%)$	$\Delta_f^a(\%)$	$t(s)$	$\Delta_f^b(\%)$	$\Delta_f^a(\%)$	$t(s)$	$\Delta_f(\%)$	$t(s)$	$\Delta_f^b(\%)$	$t(s)$	$\Delta_f(\%)$	$t(s)$
Tuzun-Burke	1.24	606.64	1.44	826.47	0.38	0.84	165.93	0.24	0.45	1620.60	0.12	0.51	2589.53	1.09	392.33	1.18	202.08	0.71	201.22
Prodhon	1.09	258.17	0.44	422.43	0.42	0.70	90.22	0.33	0.43	844.20	0.10	0.30	1162.93	0.55	176.40	0.38	191.43	0.35	91.20
Barreto	0.03	187.62	0.25	161.46	0.11	0.20	35.45	0.01	0.05	354.40	0.15	0.65	279.08	0.74	105.15	0.07	48.50	0.62	53.00
Average	**0.99**	405.35	**0.86**	563.61	**0.35**	**0.68**	115.71	**0.24**	**0.37**	1117.40	**0.12**	**0.45**	1685.38	**0.83**	263.08	**0.70**	174.36	**0.56**	135.05
Processor type	Core 2 Quad		Core 2 Quad		Opteron 275			Opteron 275			Xeon E5 462			Core 2 Duo		XP 2500+		Core 2 Duo	
Processor speed	2.83 GHz		2.66 GHz		2.20 GHz			2.20 GHz			3.00 GHz			2.00 GHz		1.83 GHz		2.00 GHz	
Passmark score	1200		1135		685			685			1219			776		546		776	
Corrected time	**294.45** $\times$ **5**		**387.22**		**47.98** $\times$ **5**			**463.33** $\times$ **5**			**1243.63** $\times$ **10**			**123.58**		**57.63** $\times$ **10**		**63.44**	

Table 1: Comparison of the results of all TBSA variants to the state-of-the-art CLRP heuristics from the literature on the Tuzun-Burke, Prodhon and Barreto benchmark sets.

In the following, we list the methods whose results are reported in Table 1, and we explain how to interpret the results and run-times of each method:

- The results of all TBSA variants are based on 5 runs, and we report the best and average solution quality, and the average time per run.
- For **DLPP** (Duhamel, Lacomme, Prins, and Prodhon 2010), the best solution obtained in 5 runs and the time of the best run are given.
- For **YLLT** (Yu, Lin, Lee, and Ting 2010), **ELT** (Escobar, Linfati, and Toth 2013), and **ELBT** (Escobar, Linfati, Baldoquin, and Toth 2014a), we provide the results of a single run.
- For **HCC** (Hemmelmayr, Cordeau, and Crainic 2012), we report the results of their ALNS using 500K and 5000K iterations. The best and average result of 5 runs, and the total run-time of the 5 runs is given. In Table 1, we translate the total run-time into an average time per run in order to be consistent with the other methods. Note that the results we use for ALNS-5000K were kindly provided by the authors HCC. In the paper by Escobar et al. (2014a), the best solutions found by the ALNS of HCC during overall testing are erroneously reported as the best of 5 runs of ALNS-5000K.
- For **CCG** (Contardo, Cordeau, and Gendron 2014a), the table provides the best and average of 10 runs and the average time per run. For **TC** (Ting and Chen 2013), only the best of 10 runs and the average run-time is reported.

Finally, we provide detailed results on an instance basis for all variants of TBSA on the Tuzun-Burke (see Table 2), Prodhon (Table 3), and the Barreto set (Table 4). Results that match the previous BKS are printed in bold, results that improve on the previous BKS are printed in bold and underlined.

3.4.1 Comparison of the different TBSA variants and effect of large composite neighborhood

Comparing the variants TBSA$_{speed}$, TBSA$_{basic}$, and TBSA$_{qual}$, we observe that we can successfully control the tradeoff between solution quality and run-time by modifying the number of iterations and the strength of the sparsification as described above. From TBSA$_{speed}$ to TBSA$_{basic}$ and from TBSA$_{basic}$ to TBSA$_{qual}$, the run-time increases by a factor of approximately 4.0, while the best (average) solution improves from a positive gap to the BKS of 0.08 (0.25) for TBSA$_{speed}$ to a negative gap of -0.06 (0.06) for TBSA$_{basic}$, and a gap of -0.08 (-0.01) for TBSA$_{qual}$. Note that in TBSA$_{qual}$, we put very strong emphasis on solution quality and are thus able to slightly improve the already very good solution quality of TBSA$_{basic}$, especially with regards to average solution quality. We observe that not only the solution quality improves from TBSA$_{speed}$ to TBSA$_{basic}$ to TBSA$_{qual}$ but also the robustness of TBSA, which is demonstrated by ever smaller differences between the best and the average solution quality.

The significant positive effect of using the large composite neighborhood described in Section 2.2.2 becomes evident from the comparison of TBSA$_{speed}$ and TBSA$_{classic}$: TBSA$_{speed}$ is able to provide better solution quality within lower run-times and thus dominates TBSA$_{classic}$. Moreover, we find that the large composite neighborhood has a clear positive impact on the robustness of TBSA.

3.4.2 Comparison to the state-of-the-art

For all variants of TBSA including TBSA$_{classic}$, the average solution quality (measured by Δ_f^b) over all three benchmark sets is superior to the solution quality of all comparison methods. It is noteworthy that even the

Inst.	BKS	TBSA$_{classic}$					TBSA$_{speed}$					TBSA$_{basic}$					TBSA$_{qual}$					$\overline{\text{TBSA}}$	
		f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	f^b	$\Delta_f^b(\%)$
111112	1467.68⋆	**1467.68**	0.00	1471.10	0.23	47.52	**1467.68**	0.00	1468.69	0.07	22.80	**1467.68**	0.00	1467.80	0.01	101.07	**1467.68**	0.00	**1467.68**	0.00	345.07	**1467.68**	0.00
111212	1394.80⋆	**1394.80**	0.00	1397.91	0.22	39.92	1396.46	0.12	1396.46	0.12	22.36	**1394.80**	0.00	1395.15	0.03	98.52	**1394.80**	0.00	1396.36	0.11	326.02	**1394.80**	0.00
112112	1167.16⋆	**1167.16**	0.00	1175.41	0.71	47.43	**1167.16**	0.00	1167.94	0.07	20.73	**1167.16**	0.00	1167.24	0.01	90.10	**1167.16**	0.00	1167.24	0.01	354.64	**1167.16**	0.00
112212	791.66⋆	791.70	0.01	793.09	0.18	29.00	791.82	0.02	794.36	0.34	16.32	**791.66**	0.00	791.96	0.04	76.13	**791.66**	0.00	**791.66**	0.00	324.80	**791.66**	0.00
113112	1238.24	1243.79	0.45	1246.34	0.65	46.30	1238.67	0.03	1239.55	0.11	19.99	1238.49	0.02	1238.74	0.04	102.83	1238.49	0.02	1238.56	0.03	349.72	1238.49	0.02
113212	902.26⋆	902.29	0.00	902.87	0.07	31.26	902.33	0.01	902.89	0.07	15.61	**902.26**	0.00	902.28	0.00	75.22	**902.26**	0.00	**902.26**	0.00	311.39	**902.26**	0.00
111122	1449.20	1453.45	0.29	1455.57	0.44	50.88	**1449.20**	0.00	1454.72	0.38	24.33	**1449.20**	0.00	1451.64	0.17	115.72	**1448.37**	-0.06	**1448.70**	-0.03	369.68	**1448.37**	-0.06
111222	1432.29	**1432.29**	0.00	**1432.29**	0.00	42.72	**1432.29**	0.00	1432.73	0.03	20.32	**1432.29**	0.00	**1432.29**	0.00	112.78	**1432.29**	0.00	**1432.29**	0.00	336.30	**1432.29**	0.00
112122	1102.24	1102.62	0.03	1105.62	0.31	50.38	**1102.24**	0.00	1102.47	0.02	16.16	**1102.24**	0.00	**1102.24**	0.00	98.97	**1102.24**	0.00	**1102.24**	0.00	354.15	**1102.24**	0.00
112222	728.30	**728.30**	0.00	728.34	0.01	38.47	**728.30**	0.00	**728.30**	0.00	12.74	**728.30**	0.00	**728.30**	0.00	74.47	**728.30**	0.00	**728.30**	0.00	294.14	**728.30**	0.00
113122	1245.30	**1245.30**	0.00	1246.16	0.07	50.19	1245.42	0.01	1246.64	0.11	18.91	1245.31	0.00	1245.78	0.04	106.57	1245.31	0.00	1245.31	0.00	390.48	**1245.30**	0.00
113222	1018.29	**1018.29**	0.00	**1018.29**	0.00	40.83	1022.79	0.44	1023.11	0.47	18.79	**1018.29**	0.00	1021.89	0.35	83.64	**1018.29**	0.00	1019.19	0.09	331.00	**1018.29**	0.00
121112	2243.40	2246.53	0.14	2261.85	0.82	241.36	2267.30	1.07	2274.98	1.41	125.30	**2240.06**	-0.15	2244.74	0.06	600.63	**2238.52**	-0.22	2246.83	0.15	1773.32	**2237.73**	-0.25
121212	2209.30	**2203.32**	-0.27	**2209.16**	-0.01	272.43	**2207.35**	-0.09	2222.58	0.60	145.34	**2202.19**	-0.32	2210.42	0.05	644.43	**2202.67**	-0.30	**2206.69**	-0.12	2110.47	**2195.17**	-0.64
122112	2073.73	2084.07	0.50	2091.38	0.85	228.92	2081.23	0.36	2083.87	0.49	116.85	**2073.28**	-0.02	2075.75	0.10	606.76	**2072.92**	-0.04	2076.19	0.12	1947.71	**2070.43**	-0.16
122212	1453.18	1458.99	0.40	1462.96	0.67	195.32	**1452.41**	-0.05	1456.09	0.20	127.69	**1452.00**	-0.08	**1452.52**	-0.05	553.85	**1449.93**	-0.22	**1450.58**	-0.18	1703.98	**1449.93**	-0.22
123112	1954.70	**1952.96**	-0.09	1959.18	0.23	234.23	1959.25	0.23	1966.38	0.60	120.39	**1943.48**	-0.57	1957.87	0.16	564.04	**1943.10**	-0.59	**1946.35**	-0.43	1746.15	**1942.23**	-0.64
123212	1762.03	1771.56	0.54	1772.79	0.61	209.60	1764.63	0.15	1766.83	0.27	103.33	**1761.80**	-0.01	1764.67	0.15	495.61	1763.20	0.07	1764.39	0.13	1565.78	**1761.11**	-0.05
121122	2138.40	2138.43	0.00	2152.32	0.65	327.39	2168.06	1.39	2174.61	1.69	107.41	2139.64	0.06	2149.93	0.54	741.00	**2138.38**	0.00	2145.87	0.35	1801.37	**2137.45**	-0.04
121222	2225.10	2228.04	0.13	2238.88	0.62	291.95	2227.98	0.13	2238.70	0.61	130.88	**2220.92**	-0.19	2235.00	0.44	660.88	**2214.86**	-0.46	**2219.90**	-0.23	2181.95	**2214.86**	-0.46
122122	1692.17	**1692.05**	-0.01	1699.23	0.42	261.17	**1685.54**	-0.39	1695.56	0.20	127.88	**1685.65**	-0.39	**1685.65**	-0.39	716.85	**1685.65**	-0.39	**1685.77**	-0.38	1974.23	**1685.52**	-0.39
122222	1082.46	1082.60	0.01	1083.59	0.10	203.55	1083.39	0.09	1086.01	0.33	100.80	**1082.46**	0.00	1083.11	0.06	563.65	**1082.46**	0.00	1082.63	0.02	1600.30	**1082.46**	0.00
123122	1926.64	**1918.76**	-0.41	**1923.58**	-0.16	236.10	**1926.54**	-0.01	1930.69	0.21	156.82	**1922.74**	-0.20	1926.76	0.01	592.04	**1910.66**	-0.83	**1915.65**	-0.57	1618.94	**1910.08**	-0.86
123222	1390.87	1392.31	0.10	1392.93	0.15	205.13	1391.87	0.07	1392.98	0.15	90.97	1391.57	0.05	1392.53	0.12	485.39	1391.46	0.04	1392.10	0.09	1484.46	**1390.86**	0.00
131112	1896.98	1897.42	0.02	1904.00	0.37	125.54	1904.28	0.38	1919.52	1.19	55.77	1897.34	0.02	1904.22	0.38	302.52	**1895.48**	-0.08	1900.59	0.19	1104.45	**1892.17**	-0.25
131212	1960.23	1968.51	0.42	1970.69	0.53	115.86	1967.80	0.39	1970.81	0.54	48.78	**1960.02**	-0.01	1966.73	0.33	281.49	**1960.02**	-0.01	1963.17	0.15	989.23	**1960.02**	-0.01
132112	1443.32⋆	1454.22	0.76	1468.93	1.77	120.67	**1443.32**	0.00	1445.19	0.13	72.00	**1443.32**	0.00	1444.71	0.10	257.53	**1443.32**	0.00	1444.20	0.06	776.99	**1443.32**	0.00
132212	1204.42	1205.62	0.10	1207.14	0.23	103.14	1205.10	0.06	1207.58	0.26	51.26	**1204.42**	0.00	1204.89	0.04	243.44	1204.62	0.02	1204.74	0.03	880.01	**1204.42**	0.00
133112	1694.18	1711.03	0.99	1714.81	1.22	112.61	**1694.18**	0.00	1708.32	0.83	49.59	1694.85	0.04	1704.36	0.60	250.61	**1694.18**	0.00	1697.10	0.17	828.29	**1694.18**	0.00
133212	1198.28	**1198.08**	-0.02	1200.17	0.16	94.19	1198.53	0.02	1199.59	0.11	42.24	**1198.08**	-0.02	1198.34	0.01	295.43	**1198.08**	-0.02	1198.60	0.03	877.64	**1197.95**	-0.03
131122	1823.53	**1822.74**	-0.04	1824.58	0.06	143.65	1824.47	0.05	1830.48	0.38	60.61	**1819.68**	-0.21	**1820.47**	-0.17	314.86	**1819.68**	-0.21	**1821.16**	-0.13	901.59	**1819.68**	-0.21
131222	1792.80	1800.19	0.41	1807.30	0.81	152.46	1798.38	0.31	1806.47	0.76	63.18	1795.84	0.17	1799.08	0.35	257.95	1795.84	0.17	1796.51	0.21	879.03	**1792.77**	0.00
132122	1434.63	1444.21	0.67	1446.25	0.81	111.18	1441.86	0.50	1447.95	0.93	72.50	**1429.42**	-0.36	1439.34	0.33	326.58	**1429.52**	-0.36	1434.95	0.02	941.75	**1429.30**	-0.37
132222	931.28	**925.46**	-0.62	**926.18**	-0.55	112.64	**925.95**	-0.57	**927.10**	-0.45	54.82	**924.95**	-0.68	**925.38**	-0.63	265.51	**925.05**	-0.67	**925.77**	-0.59	738.18	**924.68**	-0.71
133122	1392.01	1402.13	0.73	1414.53	1.62	122.30	1404.21	0.88	1410.76	1.35	55.81	1392.18	0.01	1401.11	0.65	294.79	1392.18	0.01	1399.46	0.54	882.56	**1392.01**	0.00
133222	1151.80	1152.67	0.08	1154.13	0.20	102.41	1152.49	0.06	1155.34	0.31	57.89	1151.83	0.00	1155.65	0.33	242.53	1151.83	0.00	1153.63	0.16	771.61	**1151.37**	-0.04
Average			**0.15**		**0.42**	**134.41**		**0.16**		**0.41**	**65.75**		**-0.08**		**0.12**	**324.84**		**-0.11**		**0.00**	**1004.65**		**-0.15**

Table 2: Detailed results of all variants of TBSA on the Tuzun-Burke benchmark. Results that match the previous BKS are printed in bold, results that improve on the previous BKS are printed in bold and underlined. BKS labeled with ⋆ denote optimal solutions.

Inst.	BKS	TBSA$_{classic}$					TBSA$_{speed}$					TBSA$_{basic}$					TBSA$_{qual}$					$\overline{\text{TBSA}}$	
		f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	f^b	$\Delta_f^b(\%)$
20-5-1a	54793⋆	**54793**	0.00	**54793.0**	0.00	1.40	**54793**	0.00	**54793.0**	0.00	0.80	**54793**	0.00	**54793.0**	0.00	2.79	**54793**	0.00	**54793.0**	0.00	15.84	**54793**	0.00
20-5-1b	39104⋆	**39104**	0.00	**39104.0**	0.00	0.99	**39104**	0.00	**39104.0**	0.00	0.53	**39104**	0.00	**39104.0**	0.00	2.04	**39104**	0.00	**39104.0**	0.00	11.46	**39104**	0.00
20-5-2a	48908⋆	**48908**	0.00	**48908.0**	0.00	1.40	**48908**	0.00	**48908.0**	0.00	0.74	**48908**	0.00	**48908.0**	0.00	2.82	**48908**	0.00	**48908.0**	0.00	15.53	**48908**	0.00
20-5-2b	37542⋆	**37542**	0.00	**37542.0**	0.00	1.04	**37542**	0.00	**37542.0**	0.00	0.51	**37542**	0.00	**37542.0**	0.00	2.06	**37542**	0.00	**37542.0**	0.00	11.41	**37542**	0.00
50-5-1a	90111⋆	**90111**	0.00	**90111.0**	0.00	5.05	**90111**	0.00	**90111.0**	0.00	2.48	**90111**	0.00	**90111.0**	0.00	13.37	**90111**	0.00	**90111.0**	0.00	71.22	**90111**	0.00
50-5-1b	63242⋆	**63242**	0.00	**63242.0**	0.00	4.24	**63242**	0.00	**63242.0**	0.00	2.35	**63242**	0.00	**63242.0**	0.00	10.54	**63242**	0.00	**63242.0**	0.00	63.66	**63242**	0.00
50-5-2a	88298⋆	**88298**	0.00	**88298.0**	0.00	5.37	**88298**	0.00	88344.8	0.05	3.32	**88298**	0.00	**88298.0**	0.00	14.93	**88298**	0.00	**88298.0**	0.00	82.22	**88298**	0.00
50-5-2b	67308⋆	67373	0.10	67974.6	0.99	4.20	**67308**	0.00	67399.0	0.14	3.07	**67308**	0.00	**67308.0**	0.00	14.46	**67308**	0.00	**67308.0**	0.00	81.49	**67308**	0.00
50-5-2bis	84055⋆	**84055**	0.00	84208.4	0.18	5.00	**84055**	0.00	**84055.0**	0.00	3.40	**84055**	0.00	**84055.0**	0.00	15.26	**84055**	0.00	**84055.0**	0.00	87.05	**84055**	0.00
50-5-2bbis	51822⋆	**51822**	0.00	**51822.0**	0.00	4.51	**51822**	0.00	51862.4	0.08	2.70	**51822**	0.00	**51822.0**	0.00	12.25	**51822**	0.00	**51822.0**	0.00	72.64	**51822**	0.00
50-5-3a	86203⋆	**86203**	0.00	**86203.0**	0.00	5.61	**86203**	0.00	86370.8	0.19	3.34	**86203**	0.00	**86203.0**	0.00	14.11	**86203**	0.00	**86203.0**	0.00	77.14	**86203**	0.00
50-5-3b	61830⋆	**61830**	0.00	**61830.0**	0.00	3.87	**61830**	0.00	61838.4	0.01	2.35	**61830**	0.00	**61830.0**	0.00	9.74	**61830**	0.00	**61830.0**	0.00	58.35	**61830**	0.00
100-5-1a	274814⋆	275459	0.23	275544.2	0.27	28.53	275459	0.23	275827.2	0.37	15.14	**274814**	0.00	275270.8	0.17	91.14	**274814**	0.00	**274814.0**	0.00	524.03	**274814**	0.00
100-5-1b	213568	213776	0.10	214007.4	0.21	21.07	213772	0.10	214632.8	0.50	11.68	213886	0.15	213925.2	0.17	71.21	**213568**	0.00	213598.6	0.01	441.35	**213568**	0.00
100-5-2a	193671⋆	**193671**	0.00	193794.2	0.06	17.45	**193671**	0.00	193798.6	0.07	11.86	**193671**	0.00	193762.0	0.05	69.50	**193671**	0.00	**193671.0**	0.00	383.33	**193671**	0.00
100-5-2b	157095⋆	157129	0.02	157171.8	0.05	15.86	157150	0.04	157176.2	0.05	8.11	157110	0.01	157151.2	0.04	49.83	**157095**	0.00	157142.8	0.03	319.71	**157095**	0.00
100-5-3a	200079⋆	200214	0.07	200289.4	0.11	23.72	200168	0.04	200500.8	0.21	14.05	**200079**	0.00	200165.8	0.04	91.25	**200079**	0.00	**200079.0**	0.00	460.56	**200079**	0.00
100-5-3b	152441⋆	**152441**	0.00	152752.8	0.20	17.71	**152441**	0.00	152482.2	0.03	8.39	**152441**	0.00	**152441.0**	0.00	48.53	**152441**	0.00	**152441.0**	0.00	267.60	**152441**	0.00
100-10-1a	287864	**287763**	-0.04	289083.6	0.42	34.90	287899	0.01	288346.6	0.17	25.54	**287716**	-0.05	**287806.6**	-0.02	100.21	**287668**	-0.07	**287735.8**	-0.04	486.79	**287661**	-0.07
100-10-1b	231739	232089	0.15	232689.2	0.41	21.13	231800	0.03	232067.8	0.14	16.57	**230989**	-0.32	**231551.8**	-0.08	62.57	**230989**	-0.32	**230991.6**	-0.32	318.66	**230989**	-0.32
100-10-2a	243590⋆	**243590**	0.00	243710.0	0.05	27.94	**243590**	0.00	243710.6	0.05	21.16	**243590**	0.00	243636.6	0.02	86.55	**243590**	0.00	**243590.0**	0.00	400.73	**243590**	0.00
100-10-2b	203988⋆	**203988**	0.00	204028.6	0.02	19.48	**203988**	0.00	**203988.0**	0.00	10.93	**203988**	0.00	**203988.0**	0.00	52.46	**203988**	0.00	**203988.0**	0.00	284.61	**203988**	0.00
100-10-3a	250882	251384	0.20	253656.4	1.11	34.76	252992	0.84	253208.2	0.93	22.60	**250882**	0.00	250991.4	0.04	93.67	**250882**	0.00	250947.0	0.03	473.83	**250882**	0.00
100-10-3b	204597	**203118**	-0.72	**204285.4**	-0.15	25.77	**203118**	-0.72	**203705.0**	-0.44	14.88	**203114**	-0.72	**203114.8**	-0.72	56.69	**203114**	-0.72	**203114.0**	-0.72	323.77	**203114**	-0.72
200-10-1a	475327	476702	0.29	477239.2	0.40	178.46	476845	0.32	477794.6	0.52	179.62	**475236**	-0.02	476350.6	0.22	683.59	**474920**	-0.09	**475304.0**	0.00	2522.07	**474850**	-0.10
200-10-1b	377227	**376346**	-0.23	**377220.2**	0.00	123.37	**375950**	-0.34	**376769.2**	-0.12	115.72	**375703**	-0.40	**376378.4**	-0.22	322.93	**375721**	-0.40	**376096.2**	-0.30	1536.77	**375177**	-0.54
200-10-2a	449006	**449005**	0.00	449251.0	0.05	126.82	449067	0.01	449643.8	0.14	147.04	**448378**	-0.14	**448813.2**	-0.04	581.68	**448135**	-0.19	**448181.6**	-0.18	2609.55	**448077**	-0.21
200-10-2b	374435	**374017**	-0.11	**374336.6**	-0.03	108.21	**373819**	-0.16	**374209.6**	-0.06	69.52	**373886**	-0.15	**374113.6**	-0.09	332.53	**373796**	-0.17	**373958.6**	-0.13	1639.86	**373696**	-0.20
200-10-3a	469433⋆	469717	0.06	470431.4	0.21	184.97	470909	0.31	471743.6	0.49	176.25	469493	0.01	470082.2	0.14	675.33	469461	0.01	469476.2	0.01	3726.44	**469433**	0.00
200-10-3b	362827	365031	0.61	365559.2	0.75	95.17	363189	0.10	363758.8	0.26	67.58	**362609**	-0.06	362842.8	0.00	343.21	**362599**	-0.06	**362783.6**	-0.01	1407.39	**362320**	-0.14
Average			**0.02**		**0.18**	**38.27**		**0.03**		**0.13**	**32.07**		**-0.06**		**-0.01**	**130.91**		**-0.07**		**-0.05**	**625.84**		**-0.08**

Table 3: Detailed results of all variants of TBSA on the Prodhon benchmark. Results that match the previous BKS are printed in bold, results that improve on the previous BKS are printed in bold and underlined. BKS labeled with ⋆ denote optimal solutions.

Inst.	BKS	TBSA$_{classic}$					TBSA$_{speed}$					TBSA$_{basic}$					TBSA$_{qual}$					$\overline{\text{TBSA}}$	
		f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	f^b	$\Delta_f^b(\%)$
Christofides69-50x5	565.6⋆	**565.6**	0.00	**565.6**	0.00	4.29	**565.6**	0.00	**565.6**	0.00	1.94	**565.6**	0.00	568.5	0.50	10.89	**565.6**	0.00	568.5	0.50	60.01	**565.6**	0.00
Christofides69-75x10	848.9⋆	**848.9**	0.00	853.2	0.51	22.06	**848.9**	0.00	850.5	0.19	12.14	**848.9**	0.00	**848.9**	0.00	52.64	**848.9**	0.00	**848.9**	0.00	177.60	**848.9**	0.00
Christofides69-100x10	833.4⋆	**833.4**	0.00	836.8	0.41	42.04	**833.4**	0.00	834.6	0.14	25.57	**833.4**	0.00	834.1	0.08	90.32	**833.4**	0.00	834.1	0.08	325.82	**833.4**	0.00
Daskin95-88x8	355.8	**355.8**	0.00	**355.8**	0.00	12.46	**355.8**	0.00	355.9	0.03	6.44	**355.8**	0.00	**355.8**	0.00	27.71	**355.8**	0.00	**355.8**	0.00	185.91	**355.8**	0.00
Daskin95-150x10	43952.3	44062.8	0.25	44374.3	0.96	99.78	**43919.9**	-0.07	44088.9	0.31	51.65	**43923.5**	-0.07	43981.5	0.07	190.85	**43919.9**	-0.07	**43923.5**	-0.07	772.52	**43919.9**	-0.07
Gaskell67-21x5	424.9⋆	**424.9**	0.00	**424.9**	0.00	1.11	**424.9**	0.00	**424.9**	0.00	0.65	**424.9**	0.00	**424.9**	0.00	2.30	**424.9**	0.00	**424.9**	0.00	12.74	**424.9**	0.00
Gaskell67-22x5	585.1⋆	**585.1**	0.00	**585.1**	0.00	1.20	**585.1**	0.00	**585.1**	0.00	0.47	**585.1**	0.00	**585.1**	0.00	2.04	**585.1**	0.00	**585.1**	0.00	11.19	**585.1**	0.00
Gaskell67-29x5	512.1⋆	**512.1**	0.00	515.5	0.67	1.94	**512.1**	0.00	**512.1**	0.00	0.76	**512.1**	0.00	**512.1**	0.00	3.47	**512.1**	0.00	**512.1**	0.00	20.21	**512.1**	0.00
Gaskell67-32x5-1	562.2⋆	**562.2**	0.00	564.2	0.35	2.11	**562.2**	0.00	**562.2**	0.00	1.07	**562.2**	0.00	**562.2**	0.00	4.60	**562.2**	0.00	**562.2**	0.00	23.89	**562.2**	0.00
Gaskell67-32x5-2	504.3⋆	**504.3**	0.00	**504.3**	0.00	1.70	**504.3**	0.00	**504.3**	0.00	0.78	**504.3**	0.00	**504.3**	0.00	3.48	**504.3**	0.00	**504.3**	0.00	20.12	**504.3**	0.00
Gaskell67-36x5	460.4⋆	**460.4**	0.00	**460.4**	0.00	1.83	**460.4**	0.00	**460.4**	0.00	0.87	**460.4**	0.00	**460.4**	0.00	4.47	**460.4**	0.00	**460.4**	0.00	28.78	**460.4**	0.00
Min92-27x5	3062.0⋆	**3062.0**	0.00	**3062.0**	0.00	1.69	**3062.0**	0.00	**3062.0**	0.00	0.74	**3062.0**	0.00	**3062.0**	0.00	3.14	**3062.0**	0.00	**3062.0**	0.00	17.58	**3062.0**	0.00
Min92-134x8	5709.0	5719.3	0.18	5738.3	0.51	46.42	**5709.0**	0.00	5722.2	0.23	29.11	**5709.0**	0.00	5711.0	0.04	106.22	**5709.0**	0.00	**5709.0**	0.00	572.18	**5709.0**	0.00
Average			**0.03**		**0.26**	**18.36**		**-0.01**		**0.07**	**10.17**		**-0.01**		**0.05**	**38.63**		**-0.01**		**0.04**	**171.43**		**-0.01**

Table 4: Detailed results of all variants of TBSA on the Barreto benchmark. Results that match the previous BKS are printed in bold, results that improve on the previous BKS are printed in bold and underlined. BKS labeled with ⋆ denote optimal solutions.

fastest variant TBSA$_{speed}$ is able to beat the solution quality of those methods from the literature concentrating on solution quality and having high run-times (HCC-5000K and CCG). Most important, TBSA$_{speed}$ is able to dominate all comparison methods regarding the tradeoff between solution quality and run-time. For all methods but ELT and ELBT, the run-time of all five TBSA$_{speed}$ runs is below that of the comparison method and the solution quality of TBSA$_{speed}$ based on the best of the five runs is superior as described above. For ELT and ELBT, the average solution quality of TBSA$_{speed}$ is superior and the average run-time of one run is lower. Note that dominance is not achieved without using the large composite neighborhood: Although the average solution quality of TBSA$_{classic}$ is superior to ELBT, the average run-time of one run is slightly higher (78.80 vs. 63.44 seconds).

Over all three benchmark sets, TBSA$_{basic}$ and TBSA$_{qual}$ are able to significantly improve the solution quality of all comparison methods and even show negative average gaps to the previous BKS from the literature; for TBSA$_{qual}$ even the average solution quality has a negative gap to the previous BKS. In order to achieve dominance, both variants put too much emphasis on solution quality and are thus not able to match the run-times of methods putting more emphasis on speed. Comparing the results of TBSA$_{qual}$ to the best results for each instance that were found during the overall testing of TBSA ($\overline{\text{TBSA}}$), only a small difference in solution quality can be observed. This may indicate that TBSA$_{qual}$ is quite close to the best possible solution quality for the three classical CLRP benchmark instance sets.

Concerning the performance of TBSA on an instance basis, Tables 2–4 show that:

- Of the 36 Tuzun-Burke instances, TBSA$_{classic}$ is able to match the previous BKS in 7 cases and to improve the BKS in 7 cases, TBSA$_{speed}$ matches in 8 and improves in 5 cases, TBSA$_{basic}$ matches in 13 and improves in 14 cases and TBSA$_{qual}$ matches in 13 and improves in 15 cases. During the overall testing, we were able to match 17 and improve 18 BKS.

- Of the 30 Prodhon instances, TBSA$_{classic}$ is able to match the previous BKS in 16 cases and to improve the BKS in 4 cases, TBSA$_{speed}$ matches in 16 and improves in 3 cases, TBSA$_{basic}$ matches in 19 and improves in 8 cases and TBSA$_{qual}$ matches in 21 and improves in 8 cases. During the overall testing, we were able to match 22 and improve 8 BKS.

- Of the 13 Barreto instances, TBSA$_{classic}$ is able to match the previous BKS in 11 cases, while TBSA$_{speed}$, TBSA$_{basic}$, TBSA$_{qual}$, and $\overline{\text{TBSA}}$ all match in 12 cases and improve in 1 case.

We further observe that the variants of TBSA using the large composite neighborhood demonstrate a convincing robustness measured by the differences of the gaps based on the best and average solution quality over all three benchmark sets: 0.17% for TBSA$_{speed}$, 0.12% for TBSA$_{basic}$, and 0.07% for TBSA$_{qual}$. Thus, even TBSA$_{speed}$ shows better robustness than HCC-500K and CCG (0.33%) although it has significant shorter run-times than CCG. Only HCC-5000K proves more robust than TBSA$_{speed}$ (within clearly higher run-times), but is not as robust as TBSA$_{basic}$ and TBSA$_{qual}$.

Summarizing, we note that the three variants TBSA$_{speed}$, TBSA$_{basic}$, and TBSA$_{qual}$ together form the Pareto frontier of heuristic CLRP methods, i.e., for all three variants, there is no method that achieves better solution quality within lower run-time. TBSA$_{classic}$ and all comparison methods from the literature are dominated by TBSA$_{speed}$.

3.4.3 Performance on OLRP instances

To further compare the performance of TBSA with the state-of-the-art, we use TBSA to address the OLRP introduced by Yu and Lin (2015). Contrary to the CLRP, in the OLRP, vehicles do not need to return to a

depot but can terminate their routes at any customer location. To adapt TBSA to the OLRP, we transform the given OLRP instance into a CLRP instance by setting the costs of all arcs starting at a customer vertex and ending at a depot vertex to zero. Moreover, these arcs are not considered in the diversification function Λ and in the tabu list.

In Table 5, we compare the OLRP results obtained with TBSA$_{basic}$ in 5 runs on the three classical benchmark sets described above (but interpreted as OLRP instances) with the results of the SA algorithm of **YL** (Yu and Lin 2015) obtained in a single run. Results are reported as gaps to the BKS found by YL during the overall testing of their algorithm. Reported run-times for TBSA$_{basic}$ denote the average of 5 runs, for YL the time of a single run. Table 6 presents the detailed results of TBSA$_{basic}$ on an instance basis.

TBSA$_{basic}$ proves to be very robust on the OLRP instances with a difference of only 0.1% between the best and average solution quality over all three benchmark sets. If we use Δ_f^a for the comparison to YL, TBSA$_{basic}$ is able to improve the gap to the BKS by 0.87% using not even a tenth of the runtime of YL. Thus, TBSA$_{basic}$ clearly dominates YL. Moreover, for the Tuzun-Burke set (36 instances), TBSA$_{basic}$ matches 4 and improves 25 of the previous BKS; for the Prodhon set (30 instances), it matches 14 and improves 16 BKS; for the Barreto set (13 instances), it matches 8 and improves 5 BKS.

3.5 Results on new benchmark set with focus on large-scale instances

Tables 7 and 8 report the results of TBSA$_{basic}$ on the newly generated instances described in Section 3.1. For each instance, we report the best (f^b) and average objective value (f^a) of 5 runs and the average time per run in seconds ($t(s)$). Due to a lack of comparison methods, not much can be said about the obtained solution quality, and the primary goal of reporting the detailed results lies in providing comparison values for future methods tackling our benchmark, which comprises hard-to-solve large-scale instances.

Nevertheless, we observe that TBSA$_{basic}$ exhibits robust solution behavior on the new benchmark set: the average difference between f^b and f^a over all instances is only 0.37%. Concerning scaling behavior we find that for $n \geq 300$ adding 100 customers while keeping the number of depots fixed seems to approximately double the computation time. Compared to this, increasing the number of depots by 5 while keeping the number of customers fixed has only a rather small impact on run-times. Summarizing, we may say that the average run-times, which stay below 4.5 hours for the instance group with 600 customers and 30 depots, are suitable for tackling large-scale instances of a strategic planning problem.

Benchmark	**YL**		**TBSA$_{basic}$**		
	$\Delta_f(\%)$	$t(s)$	$\Delta_f^b(\%)$	$\Delta_f^a(\%)$	$t(s)$
Tuzun-Burke	0.21	2169	-0.36	-0.19	317
Prodhon	0.60	1837	-0.98	-0.95	111
Barreto	0.12	666	-0.51	-0.45	32
Average	**0.35**	1795	**-0.62**	**-0.52**	192
Processor type	i7 2600		Xeon E5 2670		
Processor speed	3.40 GHz		2.60 GHz		
Passmark score	1918		1652		
Corrected time	**2085**		**192 $\times$ 5**		

Table 5: Overview of the results of TBSA$_{basic}$ and the comparison method by Yu and Lin (2015) on the classical CLRP benchmark sets interpreted as OLRP instances.

Tuzun-Burke							Prodhon							Baretto						
Inst.	BKS	f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	Inst.	BKS	f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$	Inst.	BKS	f^b	$\Delta_f^b(\%)$	f^a	$\Delta_f^a(\%)$	$t(s)$
111112	1121.48	**1121.48**	0.00	**1121.48**	0.00	61.41	20-5-1a	43849	**43849**	0.00	**43849.0**	0.00	4.00	Christofides69-50x5	454.95	**431.86**	-5.08	**431.86**	-5.08	12.08
111212	1051.82	**1050.59**	-0.12	**1050.84**	-0.09	58.67	20-5-1b	33564	**33564**	0.00	**33564.0**	0.00	2.91	Christofides69-75x10	604.74	**598.60**	-1.02	**598.60**	-1.02	33.57
112112	897.52	899.10	0.18	899.10	0.18	59.11	20-5-2a	41125	**41125**	0.00	**41125.0**	0.00	3.51	Christofides69-100x10	676.93	**676.63**	-0.04	681.70	0.70	69.15
112212	609.08	609.45	0.06	609.52	0.07	72.15	20-5-2b	32520	**32520**	0.00	**32520.0**	0.00	3.02	Daskin95-88x8	285.91	**285.91**	0.00	**285.91**	0.00	39.33
113112	915.88	916.05	0.02	916.90	0.11	81.73	50-5-1a	64217	**64217**	0.00	**64217.0**	0.00	16.41	Daskin95-150x10	35178.00	**35078.96**	-0.28	**35078.96**	-0.28	126.97
113212	689.11	**689.11**	0.00	689.14	0.00	72.81	50-5-1b	49114	**49114**	0.00	**49114.0**	0.00	12.94	Gaskell67-21x5	320.17	**320.17**	0.00	**320.17**	0.00	2.87
111122	1061.92	**1061.92**	0.00	1065.53	0.34	106.66	50-5-2a	68121	**68121**	0.00	**68121.0**	0.00	16.90	Gaskell67-22x5	458.33	**458.33**	0.00	**458.33**	0.00	3.11
111222	1080.23	**1076.68**	-0.33	**1079.97**	-0.02	109.29	50-5-2b	57355	**57355**	0.00	**57355.0**	0.00	19.32	Gaskell67-29x5	386.26	**386.26**	0.00	**386.26**	0.00	4.71
112122	817.73	**817.73**	0.00	818.15	0.05	125.34	50-5-2bis	60038	**60038**	0.00	**60038.0**	0.00	17.78	Gaskell67-32x5	395.57	**395.57**	0.00	**395.57**	0.00	5.61
112222	575.41	**575.41**	0.00	575.78	0.06	85.54	50-5-2bbis	41193	**41193**	0.00	**41193.0**	0.00	14.94	Gaskell67-32x5	374.70	**374.70**	0.00	**374.70**	0.00	4.72
113122	899.40	**899.40**	0.00	899.93	0.06	119.36	50-5-3a	62581	**62581**	0.00	**62581.0**	0.00	14.30	Gaskell67-36x5	399.37	**399.37**	0.00	**399.37**	0.00	2.73
113222	763.13	**763.13**	0.00	**763.13**	0.00	93.32	50-5-3b	46584	**46584**	0.00	**46584.0**	0.00	11.13	Min92-27x5	2110.03	**2110.03**	0.00	**2110.03**	0.00	3.12
121112	1676.12	**1665.33**	-0.64	**1667.83**	-0.49	483.23	100-5-1a	222634	**222455**	-0.08	**222495.2**	-0.06	79.12	Min92-134x8	4240.97	**4232.12**	-0.21	**4232.12**	-0.21	104.84
121212	1650.02	**1638.18**	-0.72	**1642.30**	-0.47	461.50	100-5-1b	189208	**188371**	-0.44	**188461.6**	-0.39	70.21							
122112	1457.20	**1440.14**	-1.17	**1445.76**	-0.79	649.18	100-5-2a	166328	**165840**	-0.29	**165930.2**	-0.24	54.18							
122212	1037.40	**1035.31**	-0.20	**1037.26**	-0.01	350.17	100-5-2b	144689	**144503**	-0.13	**144559.2**	-0.09	47.40							
123112	1433.60	**1425.15**	-0.59	**1427.60**	-0.42	577.52	100-5-3a	162746	**162062**	-0.42	**162216.4**	-0.33	69.16							
123212	1189.18	**1185.60**	-0.30	**1186.94**	-0.19	531.12	100-5-3b	134632	**134632**	0.00	**134632.0**	0.00	52.67							
121122	1671.12	**1635.33**	-2.14	**1639.91**	-1.87	746.56	100-10-1a	276859	**241160**	-12.89	**241454.2**	-12.79	88.32							
121222	1656.27	**1646.79**	-0.57	**1655.82**	-0.03	653.87	100-10-1b	220134	**209849**	-4.67	**209958.6**	-4.62	71.80							
122122	1282.16	**1266.73**	-1.20	**1272.35**	-0.77	672.73	100-10-2a	213232	**212531**	-0.33	**212668.8**	-0.26	78.55							
122222	888.13	**886.86**	-0.14	**887.31**	-0.09	492.78	100-10-2b	189818	**189818**	0.00	**189818.0**	0.00	57.13							
123122	1500.55	**1486.20**	-0.96	1501.22	0.04	707.11	100-10-3a	214056	**211201**	-1.33	**211437.6**	-1.22	95.13							
123222	1109.19	**1102.16**	-0.63	**1102.78**	-0.58	914.25	100-10-3b	187792	**183727**	-2.16	**183808.0**	-2.12	78.90							
131112	1414.86	**1407.96**	-0.49	**1413.99**	-0.06	227.48	200-10-1a	401113	**397472**	-0.91	**397583.8**	-0.88	459.60							
131212	1395.80	**1395.80**	0.00	**1395.80**	0.00	198.46	200-10-1b	342479	**335877**	-1.93	**336376.8**	-1.78	316.17							
132112	1055.13	**1054.02**	-0.11	**1054.44**	-0.07	189.22	200-10-2a	398042	**394940**	-0.78	**395052.0**	-0.75	398.04							
132212	870.46	**868.28**	-0.25	**870.10**	-0.04	174.02	200-10-2b	351707	**348355**	-0.95	**348531.4**	-0.90	356.04							
133112	1204.60	**1200.52**	-0.34	**1200.55**	-0.34	332.59	200-10-3a	391175	**386065**	-1.31	**386488.8**	-1.20	506.57							
133212	878.54	**871.49**	-0.80	**872.11**	-0.73	187.73	200-10-3b	324032	**321131**	-0.90	**321316.0**	-0.84	318.89							
131122	1368.48	**1363.36**	-0.37	**1365.25**	-0.24	337.84														
131222	1367.97	**1362.30**	-0.41	**1366.37**	-0.12	320.19														
132122	1041.07	**1037.29**	-0.36	**1037.53**	-0.34	354.43														
132222	757.57	**755.35**	-0.29	**756.21**	-0.18	256.51														
133122	1068.00	**1066.25**	-0.16	**1067.83**	-0.02	295.67														
133222	892.52	**892.49**	0.00	894.48	0.22	262.46														
Average			**-0.36**		**-0.19**	**317.28**				**-0.98**		**-0.95**	**111.17**				**-0.51**		**-0.45**	**31.75**

Table 6: Detailed results of TBSA$_{basic}$ on the classical CLRP benchmark sets interpreted as OLRP instances. Results that match the previous BKS are printed in bold, results that improve on the previous BKS are printed in bold and underlined.

Inst.	f^b	f^a	$t(s)$	Inst.	f^b	f^a	$t(s)$	Inst.	f^b	f^a	$t(s)$
								300-15-1a	856267	857193.8	2084.63
								300-15-1b	622412	624326.4	1386.62
100-5-1c	134516	134603.8	81.24	200-10-1c	156087	156771.4	589.75	300-15-1c	366770	368123.0	1613.18
100-5-1d	275749	275793.4	52.71	200-10-1d	638452	640099.0	417.52	300-15-1d	1339010	1339804.4	1556.58
100-5-1e	292311	292400.4	68.68	200-10-1e	599463	599708.2	463.54	300-15-1e	1217690	1222415.8	1949.12
								300-15-2a	759999	760278.4	2135.43
								300-15-2b	557912	559174.8	1250.03
100-5-2c	83989	84234.2	67.20	200-10-2c	144337	144374.4	476.29	300-15-2c	311558	312637.4	1465.10
100-5-2d	242266	242411.4	52.39	200-10-2d	663814	664241.2	342.97	300-15-2d	1301863	1303445.2	1578.69
100-5-2e	253888	254063.8	65.86	200-10-2e	619037	619162.0	318.10	300-15-2e	1272700	1276006.4	1853.13
								300-15-3a	778023	778382.0	1896.65
								300-15-3b	594073	595017.8	1064.88
100-5-3c	87555	87606.4	54.64	200-10-3c	184885	185913.8	588.15	300-15-3c	341712	341939.4	1434.19
100-5-3d	226783	226846.0	48.91	200-10-3d	640357	640423.4	384.69	300-15-3d	1358223	1359934.0	1495.27
100-5-3e	252603	252661.0	78.68	200-10-3e	604617	605285.6	388.10	300-15-3e	1286877	1288553.4	2208.86
100-5-4a	255853	255892.0	52.14	200-10-4a	452870	453126.6	315.54	300-15-4a	747730	748983.8	1976.41
100-5-4b	214425	214425.0	29.79	200-10-4b	369951	370228.2	215.17	300-15-4b	559877	560310.2	1291.47
100-5-4c	98129	98187.6	52.91	200-10-4c	144407	144607.8	611.48	300-15-4c	304254	304632.6	1842.92
100-5-4d	250315	250778.6	45.68	200-10-4d	618590	619015.8	369.70	300-15-4d	1288091	1289787.6	1779.44
100-5-4e	211159	211217.6	58.30	200-10-4e	562854	563419.8	403.10	300-15-4e	1173516	1175283.6	1939.34
n = 100, m = 5			**57.80**	**n = 200, m = 10**			**420.29**	**n = 300, m = 15**			**1690.10**
				200-15-1a	461203	461780.8	589.02	300-20-1a	1009840	1011300.8	1612.80
				200-15-1b	367397	367767.8	274.24	300-20-1b	739604	755196.8	1408.82
100-10-1c	92629	92683.8	98.15	200-15-1c	148218	149190.2	692.08	300-20-1c	364096	364819.4	1554.02
100-10-1d	363930	364172.0	60.95	200-15-1d	813941	814570.0	593.25	300-20-1d	1575390	1577056.8	1625.01
100-10-1e	344322	344583.0	60.10	200-15-1e	708837	719810.8	623.89	300-20-1e	1391567	1392971.0	1601.83
				200-15-2a	513893	514058.6	626.51	300-20-2a	909306	910784.2	2044.26
				200-15-2b	406843	407128.2	271.17	300-20-2b	695524	696328.8	1437.34
100-10-2c	84717	84744.2	98.74	200-15-2c	135051	135505.6	466.28	300-20-2c	299425	300442.8	1563.93
100-10-2d	343252	343252.0	55.86	200-15-2d	811722	812486.2	543.50	300-20-2d	1569139	1570655.0	1899.70
100-10-2e	332900	333181.2	60.66	200-15-2e	712524	713160.4	569.33	300-20-2e	1386386	1388265.0	2240.11
				200-15-3a	455676	456095.0	557.51	300-20-3a	929901	930748.6	1779.00
				200-15-3b	357086	357832.4	217.93	300-20-3b	751307	751886.2	1424.87
100-10-3c	85618	85711.2	97.36	200-15-3c	141129	141557.8	536.33	300-20-3c	305771	307195.0	1176.54
100-10-3d	329990	330025.2	51.56	200-15-3d	877638	878404.0	501.71	300-20-3d	1539008	1541074.6	1657.35
100-10-3e	318156	318270.2	63.86	200-15-3e	816377	817017.0	524.26	300-20-3e	1289734	1290731.6	2090.99
100-10-4a	253892	254207.4	47.77	200-15-4a	433268	433677.6	580.28	300-20-4a	859474	860141.4	1992.62
100-10-4b	211354	211358.2	39.03	200-15-4b	349269	349794.0	274.35	300-20-4b	687930	689431.6	1333.38
100-10-4c	86215	86219.0	74.05	200-15-4c	143772	144072.8	558.34	300-20-4c	300285	301125.0	1677.81
100-10-4d	328251	328344.0	63.21	200-15-4d	828144	828332.8	546.97	300-20-4d	1540194	1542447.8	1751.09
100-10-4e	308866	309298.6	71.84	200-15-4e	700202	701066.6	743.54	300-20-4e	1344056	1345433.0	2624.15
n = 100, m = 10			**67.37**	**n = 200, m = 15**			**514.52**	**n = 300, m = 20**			**1724.78**

Table 7: Detailed results of TBSA$_{basic}$ on the newly generated CLRP benchmark set.

Inst.	f^b	f^a	$t(s)$	Inst.	f^b	f^a	$t(s)$	Inst.	f^b	f^a	$t(s)$
400-20-1a	1140605	1142848.6	5459.61	500-25-1a	1773409	1774912.8	10864.49	600-30-1a	2198674	2200768.8	20140.67
400-20-1b	880393	881162.4	2844.00	500-25-1b	1331827	1334444.8	5803.00	600-30-1b	1691805	1697677.4	10673.49
400-20-1c	467755	470279.6	3935.31	500-25-1c	673495	675806.8	6000.60	600-30-1c	748714	752951.0	10822.16
400-20-1d	1956824	1960789.4	3780.24	500-25-1d	3325312	3328954.6	6627.34	600-30-1d	4213337	4217214.0	14803.88
400-20-1e	1748962	1750819.0	4669.29	500-25-1e	2971616	2975454.6	7693.69	600-30-1e	3737075	3843513.0	15500.11
400-20-2a	1053445	1055068.0	5849.92	500-25-2a	1619689	1622446.6	11998.29	600-30-2a	2017760	2019232.8	23638.81
400-20-2b	829494	830091.2	2956.62	500-25-2b	1252748	1254125.0	5212.56	600-30-2b	1602833	1604821.0	10778.67
400-20-2c	394712	396074.8	3975.54	500-25-2c	574794	576678.0	5694.01	600-30-2c	634787	638360.2	10718.97
400-20-2d	1875072	1877613.8	4029.08	500-25-2d	3338585	3340077.2	8014.10	600-30-2d	4163772	4167031.8	15024.23
400-20-2e	1608600	1610261.0	6084.98	500-25-2e	2802823	2862227.4	11992.46	600-30-2e	3682117	3714272.4	15009.62
400-20-3a	1098989	1100874.2	4704.81	500-25-3a	1725918	1728825.2	13855.28	600-30-3a	2082824	2086671.8	24897.75
400-20-3b	849555	850730.4	2730.97	500-25-3b	1305521	1306926.2	5421.25	600-30-3b	1615623	1620896.6	10761.47
400-20-3c	391928	393227.6	2898.16	500-25-3c	581425	583190.8	5098.87	600-30-3c	662569	664845.6	9549.06
400-20-3d	1929284	1930892.0	3356.00	500-25-3d	3248557	3250056.2	6323.21	600-30-3d	4068474	4071472.6	15657.11
400-20-3e	1778315	1780375.6	5251.94	500-25-3e	2768174	2795997.0	11756.12	600-30-3e	3496852	3505305.4	15179.65
400-20-4a	1081452	1083352.6	5802.08	500-25-4a	1655514	1658637.0	13964.09	600-30-4a	1940218	1943972.0	25360.71
400-20-4b	842078	845150.8	3260.79	500-25-4b	1263496	1267151.4	6661.01	600-30-4b	1560237	1561534.0	11437.18
400-20-4c	351715	352200.6	2670.50	500-25-4c	664089	667522.2	7012.11	600-30-4c	707288	708463.6	14535.48
400-20-4d	1834809	1836465.0	4318.18	500-25-4d	3362588	3364590.0	8393.65	600-30-4d	4150775	4155109.0	17602.99
400-20-4e	1620575	1621855.2	4993.24	500-25-4e	2630867	2698984.6	11932.18	600-30-4e	3622885	3629323.2	18513.18
n = 400, m = 20			**4178.56**	**n = 500, m = 25**			**8515.92**	**n = 600, m = 30**			**15530.26**
400-25-1a	1156187	1156765.4	5401.22	500-30-1a	1984150	1991223.2	11301.10				
400-25-1b	890566	892176.8	2878.20	500-30-1b	1538027	1540872.6	6369.41				
400-25-1c	395268	396632.4	3579.13	500-30-1c	614433	618107.4	6388.81				
400-25-1d	2341499	2343936.8	3281.81	500-30-1d	3742922	3745969.0	6002.25				
400-25-1e	2053366	2058418.0	4180.70	500-30-1e	3485608	3488967.8	7835.62				
400-25-2a	1091595	1092140.8	7027.44	500-30-2a	1820346	1822744.2	15907.97				
400-25-2b	869254	869963.2	3607.85	500-30-2b	1452171	1461842.2	6908.29				
400-25-2c	360923	361783.6	3497.24	500-30-2c	649471	652143.6	5547.39				
400-25-2d	2351903	2353659.2	3709.96	500-30-2d	3815306	3816129.2	8542.96				
400-25-2e	1954300	1983843.8	4964.71	500-30-2e	3293153	3344410.2	9894.94				
400-25-3a	1105783	1106974.6	5606.25	500-30-3a	1782554	1784835.0	9978.88				
400-25-3b	862180	865586.4	3395.72	500-30-3b	1422148	1424048.8	5205.87				
400-25-3c	393783	394555.4	2998.27	500-30-3c	570866	573130.6	5936.08				
400-25-3d	2321358	2324138.8	4211.50	500-30-3d	3690995	3695470.4	8057.93				
400-25-3e	1946952	1950642.4	4098.50	500-30-3e	3171977	3285552.2	10411.43				
400-25-4a	1015654	1016764.2	5539.99	500-30-4a	1716476	1722659.8	12070.89				
400-25-4b	801722	803138.4	3117.51	500-30-4b	1398401	1461336.8	6314.01				
400-25-4c	380824	381782.8	3195.05	500-30-4c	562731	565040.0	9257.42				
400-25-4d	2362571	2364869.2	4334.50	500-30-4d	3708479	3711494.6	8340.83				
400-25-4e	1992633	1993388.8	4204.16	500-30-4e	3194234	3217200.8	9918.44				
n = 400, m = 25			**4141.49**	**n = 500, m = 30**			**8509.53**				

Table 8: Detailed results of TBSA$_{basic}$ on the newly generated CLRP benchmark set.

4 Conclusion

Motivated by the challenging task of solving large-scale CLRP instances relevant in city logistics and other application contexts, we develop a new metaheuristic, called TBSA, which combines a tree-like search on depot configurations and a GTS for route improvement. The GTS uses a large composite neighborhood defined by 14 operators, which has a significant positive impact on the effectivity, efficiency and robustness of TBSA. The fastest variant, TBSA$_{speed}$, is able to dominate all previously published heuristic CLRP solution methods on the benchmark sets of Tuzun-Burke, Prodhon and Barreto, and the three variants of TBSA together form the Pareto frontier of heuristic CLRP methods. Moreover, TBSA matches or improves the large majority of previous best-known solutions on these instances. Finally, TBSA shows good scaling behavior, robustness and reasonable run-times on newly generated instances with up to 600 customers and 30 depots. Additional experiments on OLRP benchmarks confirm the convincing performance of TBSA.

Acknowledgements

We thank Vera Hemmelmayr for providing detailed computational results of the paper Hemmelmayr et al. (2012), which allowed for a meaningful comparison of solution methods.

References

M. Albareda-Sambola. Location-routing and location-arc routing. In G. Laporte, S. Nickel, F. Saldanha da Gama, and M. Albareda-Sambola, editors, *Location Science*, chapter 15, pages 399–418. Springer, 2015.

A. Alvim and E. Taillard. POPMUSIC for the world location-routing problem. *EURO Journal on Transportation and Logistics*, 2(3):231–254, 2013.

R. Baldacci, A. Mingozzi, and R. Wolfler Calvo. An exact method for the capacitated location-routing problem. *Operations Research*, 59(5):1284–1296, 2011.

S. Barreto. *Análise e Modelização de Problemas de localização-distribuição [Analysis and modelling of location-routing problems]*. PhD thesis, University of Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal, 2004.

L. Berbotto, S. García, and F. J. Nogales. A randomized granular tabu search heuristic for the split delivery vehicle routing problem. *Annals of Operations Research*, 222:153–173, 2014.

R. M. Branchini, V. A. Armentano, and A. Løkketangen. Adaptive granular local search heuristic for a dynamic vehicle routing problem. *Computers & Operations Research*, 36(11):2955–2968, 2009.

C. Contardo, J.-F. Cordeau, and B. Gendron. A GRASP + ILP-based metaheuristic for the capacitated location-routing problem. *Journal of Heuristics*, 20:1–38, 2014a.

C. Contardo, J.-F. Cordeau, and B. Gendron. An exact algorithm based on cut-and-column generation for the capacitated location-routing problem. *INFORMS Journal on Computing*, 26(1):88–102, 2014b.

J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997.

R. Cuda, G. Guastaroba, and M. G. Speranza. A survey on two-echelon routing problems. *Computers & Operations Research*, 55:185–199, 2015.

M. Drexl and M. Schneider. A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research*, 241(2):283–308, 2014a.

M. Drexl and M. Schneider. A survey of the standard location-routing problem. Working Paper LPIS-03/2014, Logistics Planning and Information Systems, TU Darmstadt, Germany, 2014b.

C. Duhamel, P. Lacomme, C. Prins, and C. Prodhon. A GRASP×ELS approach for the capacitated location routing problem. *Computers & Operations Research*, 37(11):1912–1923, 2010.

J. W. Escobar, R. Linfati, and P. Toth. A two-phase hybrid heuristic algorithm for the capacitated location-routing problem. *Computers & Operations Research*, 40(1):70–79, 2013.

J. W. Escobar, R. Linfati, M. G. Baldoquin, and P. Toth. A granular variable tabu neighborhood search for the capacitated location-routing problem. *Transportation Research Part B: Methodological*, 67:344–356, 2014a.

J. W. Escobar, R. Linfati, P. Toth, and M. G. Baldoquin. A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem. *Journal of Heuristics*, 20(5):483–509, 2014b.

V. Hemmelmayr, J.-F. Cordeau, and T. G. Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228, 2012.

J. Jin, T. G. Crainic, and A. Løkketangen. A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. *European Journal of Operational Research*, 222(3):441–451, 2012.

D. Kirchler and R. Wolfler Calvo. A granular tabu search algorithm for the dial-a-ride problem. *Transportation Research Part B: Methodological*, 56:120–135, 2013.

N. Labadie, R. Mansini, J. Melechovský, and R. Wolfler Calvo. The team orienteering problem with time windows: An LP-based granular variable neighborhood search. *European Journal of Operational Research*, 220(1):15–27, 2012.

R. B. Lopes, C. Ferreira, B. S. Santos, and S. Barreto. A taxonomical analysis, current methods and objectives on location-routing problems. *International Transactions in Operational Research*, 20(6):795–822, 2013.

R. Martinelli and C. Contardo. Exact and heuristic algorithms for capacitated vehicle routing problems with quadratic costs structure. *INFORMS Journal on Computing*, 27(4):658–676, 2015.

H. Min, V. Jayaraman, and R. Srivastava. Combined location-routing problems: A synthesis and future research directions. *European Journal of Operational Research*, 108(1):1–15, 1998.

G. Nagy and S. Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177 (2):649–672, 2007.

S. Pirkwieser and G. Raidl. Variable neighborhood search coupled with ILP-based very large neighborhood searches for the (periodic) location-routing problem. In M. Blesa, C. Blum, G. Raidl, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 6373 of *Lecture Notes in Computer Science*, pages 174–189. Springer, 2010.

D. Pisinger and S. Ropke. Large neighborhood search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 399–419. Springer, 2010.

C. Prins, C. Prodhon, and R. Wolfler Calvo. Nouveaux algorithmes pour le problème de localisation et routage sous contraintes de capacité. In A. Dolgui and S. Dauzère-Pérèz, editors, *MOSIM' 04*, volume 2, pages 1115–1122, 2004.

C. Prins, C. Prodhon, A. Ruiz, P. Soriano, and R. Wolfler Calvo. Solving the capacitated location-routing problem by a cooperative Lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41(4):470–483, 2007.

C. Prodhon and C. Prins. A survey of recent research on location-routing problems. *European Journal of Operational Research*, 238(1):1–17, 2014.

S. Salhi and G. Rand. The effect of ignoring routes when locating depots. *European Journal of Operational Research*, 39(2):150–156, 1989.

A. Stenger, M. Schneider, M. Schwind, and D. Vigo. Location routing for small package shippers with subcontracting options. *International Journal of Production Economics*, 140(2):702–712, 2012.

C.-J. Ting and C.-H. Chen. A multiple ant colony optimization algorithm for the capacitated location routing problem. *International Journal of Production Economics*, 141(1):34–44, 2013.

P. Toth and D. Vigo. The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15(4):333–346, 2003.

D. Tuzun and L. Burke. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116(1):87–99, 1999.

V. F. Yu and S.-Y. Lin. A simulated annealing heuristic for the open location-routing problem. *Computers & Operations Research*, 62:184–196, 2015.

V. F. Yu, S.-W. Lin, W. Lee, and C.-J. Ting. A simulated annealing heuristic for the capacitated location routing problem. *Computers & Industrial Engineering*, 58(2):288–299, 2010.