ALGORITHMS AND
COMPLEXITY GROUP

A Genetic Algorithm in Combination with a Solution Archive for Solving the Generalized Vehicle Routing Problem with Stochastic Demands

Benjamin Biesinger and Bin Hu and Günther Raidl
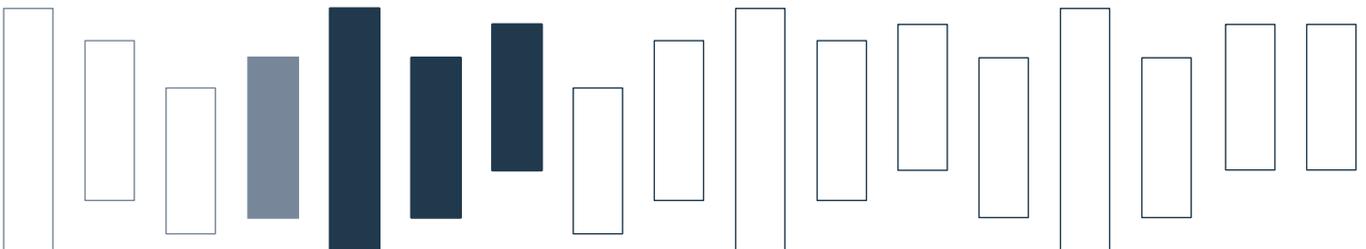
A Genetic Algorithm in Combination with a Solution Archive for Solving the Generalized Vehicle Routing Problem with Stochastic Demands

Benjamin Biesinger

Institute of Computer Graphics and Algorithms, TU Wien
Favoritenstraße 9-11/1861, 1040 Vienna, Austria, biesinger@ac.tuwien.ac.at

Bin Hu

AIT Austrian Institute of Technology
Mobility Department - Dynamic Transportation Systems
Giefinggasse 2, 1210 Vienna, Austria, bin.hu@ait.ac.at

Günther R. Raidl

Institute of Computer Graphics and Algorithms, TU Wien
Favoritenstraße 9-11/1861, 1040 Vienna, Austria, raidl@ac.tuwien.ac.at

This work presents a steady-state genetic algorithm enhanced by a complete trie-based solution archive for solving the generalized vehicle routing problem with stochastic demands using a preventive restocking strategy. As the necessary dynamic programming algorithm for the solution evaluation is very time consuming, considered candidate solutions are stored in the solution archive. It acts as complete memory of the search history, avoids re-evaluations of duplicate solution candidates and is able to efficiently transform them into guaranteed new ones. This increases the diversity of the population and reduces the risk of premature convergence. Similar to a branch-and-bound algorithm, the tree structure of the solution archive is further exploited to compute lower bounds on the nodes to cut off parts of the solution space which evidently do not contain good solutions. Since in each iteration a not yet considered solution candidate is generated and completeness can be efficiently checked, the overall method is in principle an exact enumeration algorithm, which leads to guaranteed optimal solutions for smaller instances. Computational results of this algorithm show the superiority over the so far state-of-the-art metaheuristic.

Key words: stochastic vehicle routing problem; generalized vehicle routing problem; genetic algorithm, complete solution archive

1. Introduction

Vehicle Routing Problems (VRPs) are among the most important and widely studied transportation and logistics problems in the field of combinatorial optimization. In the classical variants a set of delivery or pick-up routes for a capacity constrained fleet of vehicles starting from a central depot has to be designed in order to satisfy customers' demands. In this work we consider two generalizations of this basic problem:

- In some applications specific delivery locations are not of importance but requested goods can be brought to any delivery points in the surrounding areas of the customers. In practice, the redistribu-

tion within each area is then carried out by the customers. Practical examples of this generalization are disaster relief operations to distribute medical staff or equipment to damaged sites (Afsar et al. 2014) and the distribution of goods over sea to a number of customers in an archipelago, where each island has several ports from which the actual point of delivery can be chosen (Ghiani and Improta 2000). Ghiani and Improta (2000) originally introduced this VRP variant and named it Generalized Vehicle Routing Problem.

• The actual demand of the customers may not be precisely known in advance, resulting in the vehicle routing problem with stochastic demands (VRPSD). This situation can occur in urban waste collection, where garbage trucks need to collect the waste from certain collection points to deliver it to a central landfill site (Yang et al. 2000), or in the delivery of petrol to petrol stations (Bianchi et al. 2006). In practical applications the demands are usually not uniformly random but specific probability distributions can be deduced from historical data.

The generalized vehicle routing problem with stochastic demands (GVRPSD), which is considered in this work and has been introduced in (Biesinger et al. 2015b,c), considers both above extensions at once. A cluster of delivery points is given for each customer, as well as a stochastic demand, which is modeled by a random variable with a certain probability distribution. The aim of this problem is to plan so-called *a-priori* routes with minimum expected length or costs, respectively.

An important characteristic of stochastic routing problems is that the planned routes may not be followed as planned. Since the demand of the visited clusters may be higher than expected the vehicle may be depleted before the tour is finished. Then, a recourse action must be executed in order to satisfy the remaining demand of the tour. The most widely used recourse action in the literature, which we call *standard* restocking henceforth, sends the vehicle back to the depot whenever it is not able to service a current customer, e.g., Bertsimas (1992), Laporte and Louveaux (1998), Gendreau et al. (1996), Hjorring and Holt (1999), Rei et al. (2010). However, this strategy is sub-optimal with respect to the expected length of the routes as shown by Yang et al. (2000). A recourse action which may result in shorter routes is the *preventive* restocking strategy which allows return trips to the depot before the vehicle is fully depleted. Although expected costs can frequently be significantly lower by employing such a strategy, the computational overhead for computing them is substantial. A dynamic programming algorithm can be used for this purpose. In this work we consider such a preventive restocking policy and a relatively efficient computation of the expected costs is explained in Section 4.1.

Since this problem is a generalization of the classical VRP, it is NP-hard and also tough to solve in practice. An existing exact algorithm is limited to small instances. In this work a genetic algorithm with solution archive (GASA) and variable neighborhood descent (VND) is introduced which solves the GVRPSD heuristically. The main feature of GASA is the complete solution archive, which

efficiently detects already generated solution candidates and transforms them into new, yet unvisited ones. The solution archive combines the heuristic search with an in principle complete, tree search approach which is further exploited by computing dual bounds in the nodes to cut off solution sub-spaces that evidently do not contain promising solution candidates.

The main contributions of this article are thus: (1) A new combination of tree search methods with genetic algorithms in the context of problems with permutation encodings; (2) efficiently reducing the search space by applying bounding and avoiding search space areas of inferior solution candidates; (3) a state-of-the-art metaheuristic for a rather new variant of the vehicle routing problem.

This article is structured as follows. In Section 2 a formal problem description of the GVRPSD under the preventive stocking strategy is presented. Section 3 gives an overview of the related and previous work to the GVRPSD and in Section 4 the developed algorithm is described. A computational study is presented in Section 5, after which conclusions are drawn and an outlook is given in Section 6.

2. Problem Definition

The GVRPSD is defined on a complete undirected graph $G = (V, E)$ with node set V and edge set E. The nodes are partitioned into disjoint clusters $C = \{C_0, C_1, \ldots, C_m\}$, $C_i \subseteq V$, $\forall i = 0, \ldots, m$, such that $C_0 \cup C_1 \cup \cdots \cup C_m = V$. Each edge $(i, j) \in E$ has a distance or cost value $d_{ij} \geq 0$. Node v_0 represents the depot node and is the only node contained in C_0. Each other cluster C_j, $j = 1, \ldots, m$ has an associated stochastic demand ξ_j which is modeled as a random variable with a known discrete probability distribution and has r possible values $\xi_j^1, \ldots, \xi_j^r$. Thus, we know for each cluster C_j the probability mass function given by values p_{jk} for all $k = 0, \ldots, Q$ denoting the probability that cluster j has an actual demand of k. Furthermore, we are given one vehicle with a limited capacity Q. Situations where the demand exceeds the vehicle capacity are not considered, so we assume that $p_{jk} = 0$, $\forall j = 1, \ldots, m$, $\forall k > Q$. The goal is to find a tour starting from the depot which visits one node from each cluster exactly once and returns to the depot with minimum expected costs. During the route the clusters' actual demands, which depend on the realization of the random variables ξ_j, get revealed upon arrival and the load of the vehicle reduces by exactly these amounts. Intermediate visits of the depot are always allowed and become necessary when the vehicle cannot satisfy the demand of a cluster. Note that without further restrictions the planning of only one tour is sufficient because by employing the preventive restocking strategy the capacity constraints cannot be violated as the restocking trips are dynamically planned.

Figure 1 shows an example of a solution candidate for a small instance. In this example the vehicle capacity $Q = 10$, and all clusters except C_1 have a constant demand of 6 for cluster C_4 and 1 for the other clusters C_2, C_3, and C_5. Depending on the realization of ξ_1 a tour without an intermediate
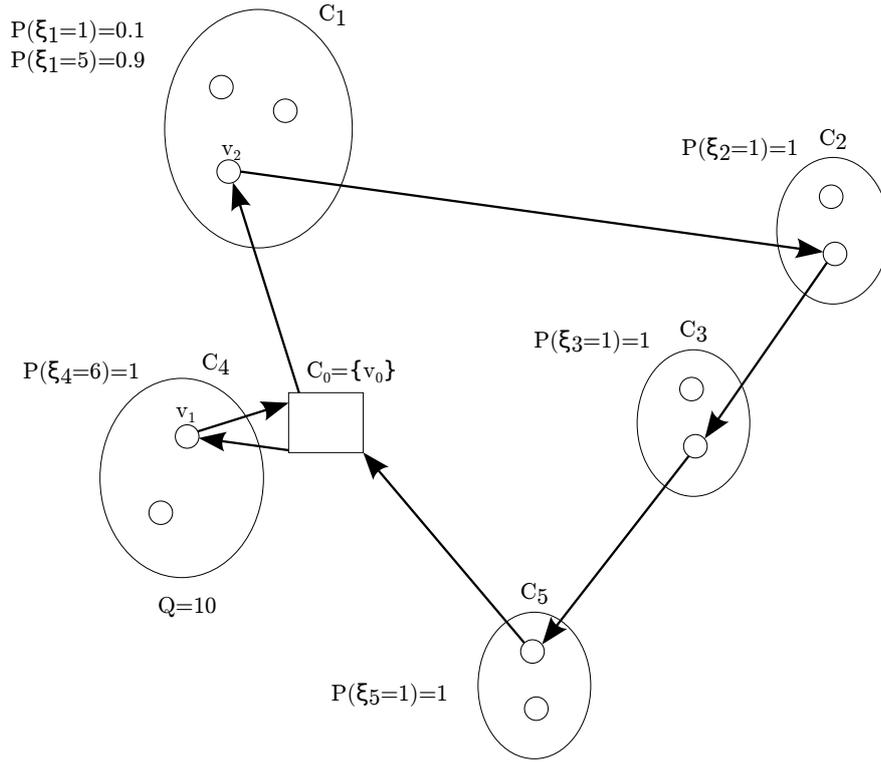
Figure 1 **Example of a solution for an instance of the GVRPSD.**

return to the depot could be planned (if $\xi_1 = 1$) or a restocking has to be performed (if $\xi_1 = 5$). However, as the actual demand only becomes known upon arrival at C_1 a restocking trip back to the depot would be needed with a high probability of 0.9. Therefore, as we use the *preventive* restocking strategy an anticipatory restocking trip from v_1 back to the depot v_0 is beneficial because its cost is significantly lower than the cost of the likely needed restocking trip from v_2.

3. Literature Survey

As the generalized vehicle routing problem with stochastic demands is a relatively new variant of a VRP, there is not much specific literature available yet. It was introduced by Biesinger et al. (2015b,c) who presented an initial attempt to solve small instances of the problem with up to 40 nodes and 14 clusters exactly by using an integer L-shaped method (Biesinger et al. 2015b) and a variable neighborhood search to tackle larger instances with up to 101 nodes and 34 clusters (Biesinger et al. 2015c). The authors also presented a multi-level evaluation scheme which significantly reduces the time needed for the solution evaluations.

When each generalization is considered separately, the literature for the GVRP and the VRPSD is richer. Since the introduction of the GVRP by Ghiani and Improta (2000), several exact and heuristic methods have been proposed for solving the problem. Exact methods include the compact mixed integer programming formulations by Kara and Bektaş (2003), by which they solved instances with

up to 50 nodes and 25 clusters. More elaborate exact methods include branch-and-cut algorithms by Bektaş et al. (2011) and Hà et al. (2014), with the latter being based on a two-commodity flow model, and a column generation approach by Afsar et al. (2014). The latter presents also two heuristics based on a route-first, cluster-second approach, in which the split procedure is executed using an iterated local search. Other heuristic methods for the GVRP include a genetic algorithm (Pop et al. 2010), a variable neighborhood search (Pop et al. 2014), and a hybrid metaheuristic combining a greedy randomized adaptive search procedure with an evolutionary local search (Hà et al. 2014). The largest instance which is tackled by all of these algorithms contains 262 vertices and 131 clusters.

In the area of vehicle routing problems with stochastic demands most works use the standard restocking approach. There is much literature for exact methods, e.g., Gendreau et al. (1995), Hjorring and Holt (1999), Laporte et al. (2002), Christiansen and Lysgaard (2007), Jabali et al. (2014) and heuristic approaches, e.g., Gendreau et al. (1996), Rei et al. (2010), Goodson et al. (2012). Current state-of-the-art exact solution approaches based on the integer L-shaped method are able to solve instances with up to 100 customers and 2 vehicles (Laporte et al. 2002) or instances with up to 4 vehicles but only 60 customers (Jabali et al. 2014). Christiansen and Lysgaard (2007) complement the L-shaped method by introducing a branch-and-price algorithm for solving the VRPSD and are able to solve instances with up to 60 customers and 16 vehicles with tighter capacity constraints.

The situation changes when preventive restocking is considered. To the best of our knowledge there is no exact algorithm described for the VRPSD with preventive restocking. However, the L-shaped method for the GVRPSD with preventive restocking from Biesinger et al. (2015b) can also be used to solve the non-generalized version. Several authors developed metaheuristics. Yang et al. (2000) were the first to introduce the preventive restocking strategy and a dynamic programming (DP) procedure to compute the expected costs of a tour using this strategy. The authors also describe two heuristics for solving a variant of the VRPSD in which the maximum planned expected route length is limited. Based on that DP Bianchi et al. (2006) developed several metaheuristics and approximative algorithms for move evaluations in the Or-opt local search neighborhood structure. Marinakis et al. presented a particle swarm optimization (PSO) algorithm (Marinakis et al. 2013), extended it with a combinatorial expanding neighborhood topology (CENTPSO) (Marinakis and Marinaki 2013) and a memetic differential evolution algorithm (Marinakis et al. 2015) for solving the VRPSD with preventive restocking. The latest heuristic solution method is a glowworm swarm optimization which makes use of path relinking and a variable neighborhood search (Marinaki and Marinakis 2015) which, together with a hybrid clonal selection algorithm (Marinakis et al. 2014) and the CENTPSO, constitute the current state-of-the-art algorithms.

4.   Algorithm Description

This section describes the proposed genetic algorithm with solution archive (GASA) for the GVRPSD. We note here that this algorithm is also suited to solve the VRPSD and computational tests have been performed for this problem as well; they will be shown in Section 5. The overall algorithmic framework, which also uses a variable neighborhood descent (VND) procedure, is depicted in Algorithm 1.

Algorithm GASA

begin

   Initialize Population;

   while unconsidered solutions remaining according to the solution archive

      Select parent solutions x_{P_1} and x_{P_2};

      Derive child x_C from x_{P_1} and x_{P_2} using a crossover operator;

      Perform mutation of x_C with probability p_{mut};

      if $f(x_C) < \alpha f(x_{\mathrm{best}})$ **then**

         Improve x_C by executing VND(x_C);

      else

         if x_C is not yet contained in the solution archive **then**

            Insert x_C into the solution archive;

         else

            Convert x_C;

      Delete the worst individual of the population;

      Add x_C to the population;

   end while

   Return best found solution;

end

Algorithm 1: Genetic algorithm with solution archive

The genetic algorithm is a steady-state GA, which replaces in each iteration the worst solution of the current population with the newly created solution. Within the GA a VND procedure is executed for promising solution candidates whose objective value, which is computed by the function $f(\cdot)$, is close to the best solution found so far, where closeness is defined by the parameter α. After the genetic operators produced the new solution candidate, it is either inserted into the solution archive or converted if the archive already contains the new solution. This step is skipped when VND was performed on this solution because then the insertion / conversion procedure is carried out within the VND. The algorithm terminates after a specific time limit T_{max}.

The individual components of GASA are described in the following subsections. First, in Section 4.1, the solution representation and the corresponding solution evaluation method is described. Section 4.2 addresses the framework of the GA which includes the initial population generation

method and its operators. The VND is presented in Section 4.3 and the solution archive with its bounding extension is explained in Section 4.4.

4.1. Solution Representation and Evaluation

Each solution candidate is represented by a permutation of the clusters which encodes the sequence of the clusters visited in the tour excluding the depot cluster C_0 at the beginning and end. To this cluster representation a dynamic programming (DP) algorithm is applied for computing the expected costs of the tour using the preventive restocking strategy. The DP, which runs in $\mathcal{O}(|V|\,Q^2)$ time, is based on the approach by Biesinger et al. (2015c) and adapted from the DP for the VRPSD (Yang et al. 2000).

Before the DP recursion is presented we explain the used notation. Function $f_{ij}(q)$ is defined for all $q = 0, \ldots, Q$, $j = 0, \ldots, m$, $i = 1, \ldots, |C_j|$ and represents the remaining cost of the tour after servicing the i-th node of cluster j with the residual vehicle capacity q. The auxiliary function $b_j(l)$ returns the l-th node of cluster j, $\forall j = 0, \ldots, m$, $l = 1, \ldots, |C_j|$. When we assume that the cluster permutation is relabeled such that the tour is $t = (C_1, \ldots, C_m)$, the DP algorithm is given by the following recursion:

$$f_{ij}(q) = \min\{f_{ij}^p(q), f_{ij}^r(q)\}$$

$$
\begin{aligned}
f_{ij}^p(q) = \min_{l=1,\ldots,|C_{j+1}|}\Big\{ d_{b_j(i),b_{j+1}(l)} &\quad + \sum_{k:\xi_{j+1}^k \leq q} f_{l,j+1}(q - \xi_{j+1}^k) p_{j+1,\xi_{j+1}^k} \\
&\quad + \sum_{k:\xi_{j+1}^k > q} \Big[2 d_{b_{j+1}(l),0} + f_{l,j+1}(q + Q - k) \Big] p_{j+1,k} \Big\}
\end{aligned}
$$

$$
f_{ij}^r(q) = d_{b_j(i),0} + \min_{l=1,\ldots,|C_{j+1}|}\Big\{ d_{0,b_{j+1}(l)} \quad + \sum_{k=1}^{r} f_{l,j+1}(Q - \xi_{j+1}^k) p_{j+1,\xi_{j+1}^k} \Big\}
$$

$$\forall q = 0, \ldots, Q, j = 0, \ldots, m, i = 0, \ldots, |C_j|$$

with the boundary condition

$$f_{im}(q) = d_{b_m(i),0}, \quad \forall q = 0, \ldots, Q, \ i = 0, \ldots, |C_m|$$

This algorithm computes for each node i and each vehicle load q if it is more cost-efficient to proceed directly to the next cluster with cost $f_{ij}^p(q)$ or to perform a preventive restock with cost $f_{ij}^r(q)$. The total expected cost of t is then given by $f_{0,0}(Q)$.

Such an expensive evaluation procedure is inconvenient and therefore Biesinger et al. (2015c) suggest a multi-level evaluation scheme (ML-ES) procedure to significantly reduce the run-time for solution evaluations within the presented metaheuristic. We apply this ML-ES also here. Its basic principle is to scale down Q and to adapt the probability distributions of the clusters accordingly to

get a fast approximation of the exact objective value. Starting from Q on each level of approximation the vehicle capacity is divided by two and rounded to the upper integer, which reduces the time for evaluation by approximately a factor of four. This can be repeated up to $\lceil \log_2 Q \rceil$ times so that on the last level $Q = 1$ and therefore the worst case complexity of the run-time for evaluation reduces to $\mathcal{O}(n)$. Biesinger et al. (2015c) also showed that the approximated objective value on level $i \in \{1, \ldots, \lceil \log_2 Q \rceil\}$ is a lower bound to the objective value of level $i - 1$. By using this theorem the iterative evaluation procedure ML-ES starts by evaluating at the highest approximation level and iteratively continues with the next lower level until it is either shown that

- the solution candidate is worse than the best solution found so far and thus can be discarded or
- an exact evaluation is performed.

Algorithm ML-ES($t, bestObj$)
begin
    $obj = 0$
    $i = \lceil \log_2 Q \rceil$
    while $obj < bestObj \wedge i \geq 0$
        $Q^i = \lceil \frac{Q}{2^i} \rceil$
        for all $j = 1, \ldots, m$ and $k = 0, \ldots, Q^i$
        $p_{jk}^i = \sum\limits_{l=2^i k}^{2^i k + 2^i - 1} p_{jl}$
        $obj = DP(t, Q^i, p^i)$
        $i = i - 1$
    Return obj
end

Algorithm 2: Multi-level evaluation scheme

Algorithm 2 shows the ML-ES in pseudocode. In each iteration of the ML-ES the dynamic programming algorithm is executed on tour t with the adapted vehicle capacity Q^i and the adapted probability distribution p^i. Note that the scaled down probability distributions do not have to be computed during each solution evaluation but can precomputed once at the start of the algorithm.

4.2. Genetic Algorithm

The proposed genetic algorithm has a fixed population size of P_{size} and uses several types of solution construction heuristics and genetic operators which are described in the following.

4.2.1. Initial Population
The choice of the generation method for the initial population of this GA is important and the aim here is to include both diverse and high quality individuals. Therefore, three different methods for solution initialization are applied with specific particular purpose:

1. **High quality:** To get one initial solution candidate of typically relatively high quality we solve the generalized travelling salesman problem (GTSP) with the given instance ignoring the demands as underlying graph. The Integer Linear Programming (ILP) model by Fischetti et al. (1997), which is based on an undirected cut-set formulation, is solved with a branch-and-cut algorithm based on CPLEX. After an optimal solution to this model is obtained a VND is performed starting from this solution to obtain a typically even better initial solution candidate; see Section 4.3 for a detailed description of the VND. Note that due to the relatively high computational effort for solving the ILP, this solution generation method is aborted after 120 seconds with the best solution found so far. If no solution could be obtained within that time a randomly generated solution is used instead.

2. **Medium quality / medium diversity:** For Euclidean instances the next $\lfloor \frac{P_{\text{size}}-1}{2} \rfloor$ initial solutions are generated by using a *farthest insertion* heuristic based on cluster distances. We compute the distances between every pair of two clusters by taking the Euclidean distances between their geometric centers, which are obtained by taking the arithmetic mean of the Euclidean coordinates of their nodes. Then a starting cluster is chosen at random and the other clusters are iteratively inserted at the best possible position of the current tour by always taking the farthest, not yet inserted cluster from the last inserted one. Ties are broken randomly.

3. **Low quality / high diversity:** The remaining $\lceil \frac{P_{\text{size}}-1}{2} \rceil$, or $P_{\text{size}} - 1$ for non-Euclidean instances, individuals are generated uniformly at random.

4.2.2. Genetic Operators For selecting the crossover candidates a tournament selection is employed. The GA uses a cyclic crossover operator to generate one child solution out of two parent solutions (A and B). This operator takes a randomly chosen sub-tour of the parent A and successively appends clusters from parent B starting from the last node of the sub-tour, skipping any already considered clusters. For diversification a *swap*-mutation operator is developed which swaps two randomly chosen cluster positions. This move is repeated for *nMut* times, where *nMut* is a parameter of the algorithm.

4.3. Variable Neighborhood Descent

To intensify the search a variable neighborhood descent (VND) algorithm with four different neighborhood structures is used:

$\mathcal{N}_1$ *1-shift*: One cluster is shifted to another position.

$\mathcal{N}_2$ *2-opt*: A sequence of clusters between two positions is inversed.

$\mathcal{N}_3$ *Or-opt*: Two or three consecutive clusters are shifted to another position.

$\mathcal{N}_4$ *SAConv*: One solution conversion based on the solution archive is performed.

The VND is executed during the GA for each solution candidate x whose objective value is at most α times larger than the best solution found so far where α is an exogenous parameter. The

first three neighborhood structures are well-known for routing problems with permuation encoding and are also used within the variable neighborhood search by Biesinger et al. (2015c). The fourth is new and based on the solution archive, which is described in detail in Section 4.4. Having defined the neighborhood structures, the complete VND with the solution archive is shown in Algorithm 3 as pseudcode.

procedure VND(x)
begin
  $l \leftarrow 1$;
  repeat
    $x^* = x$;
    $f^* = f(x)$;
    for all $x' \in \mathcal{N}_l(x)$
      if x' is already contained in the solution archive **then**
        if SA_conv **then** Convert x';
        else continue;
      else
        Insert x' into the solution archive;
      if $f(x') < f^*$ **then**
        $f^* = f(x')$;
        $x^* = x'$;
    if $f^* < f(x)$ **then**
      $x \leftarrow x^*$;
      $l \leftarrow 1$;
    else
      $l \leftarrow l + 1$;
  until $l > l_\mathrm{max}$;
  return x;
end

Algorithm 3: Variable neighborhood descent with solution archive

The VND systematically searches the given neighborhoods and basically follows the standard procedure as described by Hansen and Mladenović (2001) using a best improvement step function for $\mathcal{N}_1$ to $\mathcal{N}_3$. However, before each neighboring solution is evaluated it is checked if it is already contained in the solution archive. Depending on a binary parameter SA_conv this solution is either converted into a new solution or its evaluation is skipped and the search continues as if it is already contained in the archive.

4.4. Solution Archive

An important part of the presented genetic algorithm is the employed solution archive (SA). Complete trie-based solution archives have been introduced for evolutionary algorithms by Raidl and Hu (2010). Such an archive stores all generated solution candidates in order to efficiently detect duplicate solutions which are subsequently converted into guaranteed new and usually similar ones. It has been shown that duplicates in genetic algorithms frequently have a negative effect on the performance (Ronald 1998, Mauldin 1984) and therefore adding them to the population should be avoided. As shown in Algorithm 1 and 3 the SA is used in all parts of the algorithm and is attached to the GA after mutation and after each neighborhood move in the VND. Several applications to different kinds of combinatorial optimization problems showed that especially for problems that have a costly solution evaluation and a compact representation, which we are exactly facing with the GVRPSD, such a complete solution archive is frequently able to boost the performance of a pure genetic algorithm significantly. Examples of such problems where solution archives have already been successfully applied are benchmark problems like NK landscapes and Royal Road functions (Raidl and Hu 2010) but also more practical relevant problems like the generalized minimum spanning tree problem (Hu and Raidl 2012a,b), the rooted delay-constrained minimum spanning tree problem (Ruthmair and Raidl 2012) and several variants of competitive facility location problems (Biesinger et al. 2015a, 2014a,b).

4.4.1. Trie Structure The underlying data structure of the solution archive is a trie, which is a tree data structure often used for storing strings, e.g., in natural language dictionary applications or for string retrieval in general (Stephen 1994). In contrast to those applications where the look-up time is crucial and the stored strings are more or less fixed, in our application this data structure is highly dynamic with a lot of insertions, searches and conversions. Using an indexed trie each of these operations can be performed efficiently, i.e., independently of the number of contained elements.

In Figure 2 the trie structure for the proposed solution archive is shown. Each level i represents a position in the permutation based solution representation whereas each trie node of a level corresponds to a specific variable assignment of the first i positions. The size of each trie node is decreasing with increasing depth and has m possible child nodes on the first level. Each node q on level l has the same structure consisting of $m - l + 1$ entries $q[0], \ldots, q[m - l + 1]$. Each entry can either be a pointer to another trie node on level $l + 1$ (denoted by an arrow), a *null*-pointer (denoted by a slash), or a *complete*-pointer (denoted by a C). Now let $(i_1, \ldots, i_m)$ be the cluster permutation representing a solution candidate which should be inserted. Then, each variable i_l is related to a node q of level l in the trie. This node q splits the solution space into $m - l + 1$ parts, where in all subspaces the variables i_1 to i_{l-1} are fixed according to the path from the root node to q. Figure 2 shows two already inserted
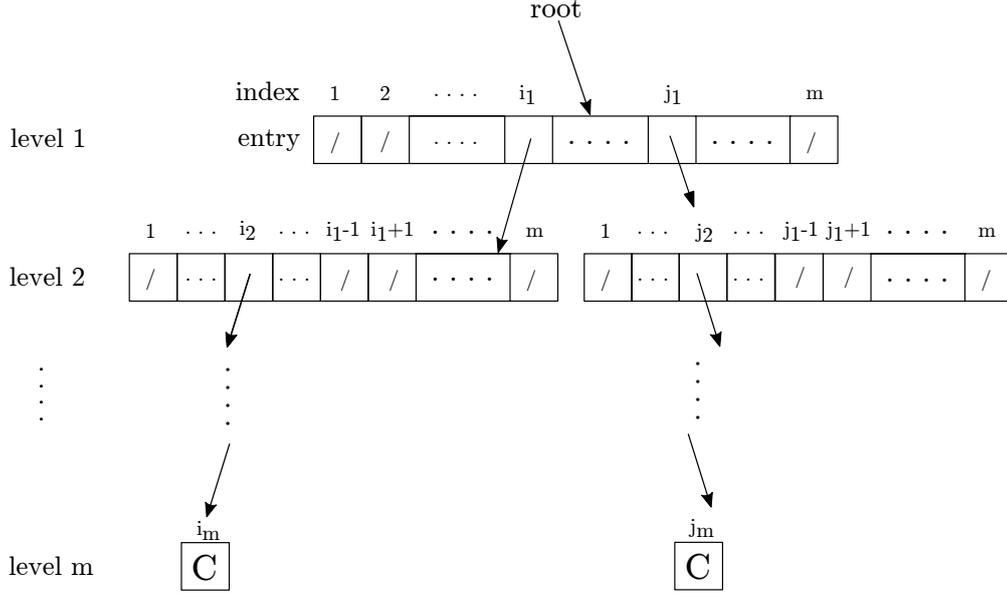
Figure 2 **Solution archive with two solution candidates** $(i_1, i_2, \ldots, i_m)$ **and** $(j_1, j_2, \ldots, j_m)$.

solution candidates $(i_1, \ldots, i_m)$ and $(j_1, \ldots, j_m)$. Here we see, that the decision of the node on the first level i_1 or j_1, respectively, fixes the first variable so that on the second level only $m-1$ decisions remain. This number of decisions, which corresponds to deciding which cluster is visited next in the sequence, decreases on each level so that on level m the last decision, i.e., the last remaining cluster, is already fixed. Therefore a *complete*-pointer is set to the associated entry, which denotes that all permutations of the corresponding subtrie have already been considered. Contrary, a *null*-pointer denotes that this is a yet completely unexplored subspace. In order to reduce the memory consumption of such a solution archive, trie nodes which exclusively contain *complete*-pointer are deleted and the corresponding parent pointer is set to *complete*.

This structure is similar to an explicitly stored branch-and-bound tree which is further exploited in Section 4.4.3 when lower bounds on partial solutions are computed to cut off subtries which evidently cannot contain good solution candidates.

4.4.2. Solution Conversion Whenever the insertion procedure detects a duplicate solution a conversion is performed. Assume that the solution $x = (x_1, \ldots, x_m)$ is inserted and on level $l \in \{1, \ldots, m\}$ a *complete*-pointer is encountered. Let $P = \{q_1, \ldots, q_l\}$ be the trie nodes visited during the insertion. Then, a conversion is performed by choosing a conversion node $q' \in P$ randomly which has at least one other entry whose value is not a *complete*-pointer. If there is no such node we know that the whole solution space has been covered and we can stop the optimization with the so far best solution candidate being a proven optimum. Otherwise we pick a non-*complete* entry $q'[k]$, $k \in \{0, \ldots, m-l+1\}$ uniformly at random and swap its index x'_l with x_l in the solution. If the value at $q'[k]$ was a *null*-pointer we know that this new solution obtained by the swap has not been considered
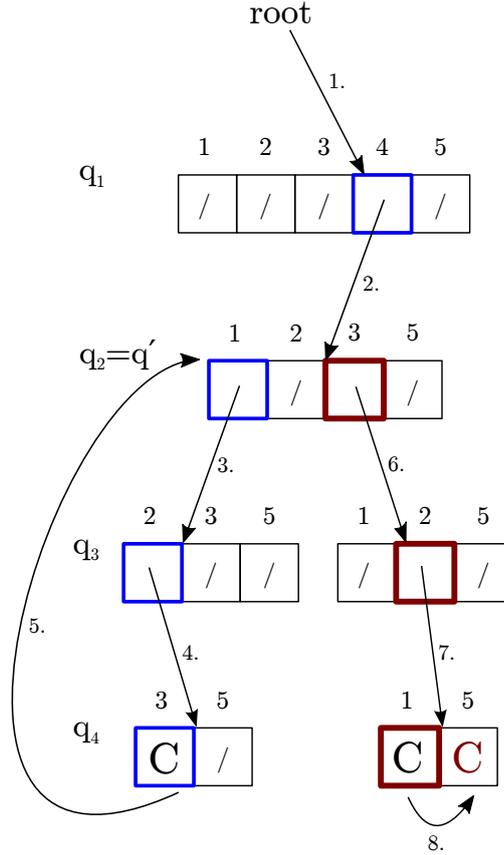
Figure 3   Example of a conversion operation in the solution archive transforming the duplicate solution $(C_4, C_1, C_2, C_3, C_5)$ **into the new solution candidate** $(C_4, C_3, C_2, C_5, C_1)$**.**

so far and therefore we insert it from node q on, which completes the conversion. Otherwise if the value at $q'[k]$ was a pointer to another trie node we could end up in a *complete*-pointer again. Then, analogously, another swap is performed. This procedure is repeated until level m is reached, at which point a guaranteed new and usually similar solution after at most $m - l$ swaps has been derived.

An example of a solution conversion for an instance with five clusters is shown in Figure 3. The sequence of visited trie nodes is denoted by the enumeration of the arcs starting from the root node and ending at the node where the conversion ends. The solution archive contains already two solutions $s_1 = (C_4, C_1, C_2, C_3, C_5)$ and $s_2 = (C_4, C_3, C_2, C_1, C_5)$ before the duplicate s_1 is inserted again into the archive by following the first four arcs. On node q_4 the duplicate is detected and consequently a conversion is performed. The node q_2 is chosen for conversion among all the visited nodes $\{q_1, \ldots, q_4\}$. On that level a swap of C_1 and C_3 is performed leading to the intermediate solution s_2. However, while inserting the remaining solution it is observed that it is still not a new solution yet, so another conversion on level 4 has to be performed leading to the final converted solution candidate $(C_4, C_3, C_2, C_5, C_1)$.

4.4.3.   Computing Lower Bounds for Partial Solutions As an additional feature of the solution archive a bounding extension is added, which is similar to the one described by Hu and Raidl (2012b) for the generalized minimum spanning tree problem. It is based on one of the basic ideas of a tree search like branch-and-bound: as mentioned in Section 4.4.1 each node of the trie represents a subspace of all solutions. If meaningful lower bounds for the objective values of the solutions associated with trie nodes can be computed, some of these nodes can likely be pruned in a branch-and-bound manner.

Before we compute lower bounds on trie nodes, we reverse the order of the variables as they are considered in the trie, i.e., for a given solution $x = (x_1, \ldots, x_m)$ the variable order is $x_m, x_{m-1}, \ldots, x_1$. This order is beneficial for the bound computation as we will see next. To compute a lower bound for a particular subtrie represented by entry k of a trie node q on level l, we partition the set of clusters into three disjoint subsets C_0, C^f, and C^o as shown in Figure 4. C^f denotes the set of fixed clusters, which is given by the fixed part of the solution $(x_m, \ldots, x_{m-l+1})$. Assume that the last fixed cluster of C^f is cluster C_l. $C^o = C \setminus (C^f \cup C_0)$ is the set of open clusters, for which the sequence of visit is still unknown. For these clusters four conditions are relaxed:

1. Capacity constraints of the vehicle.

2. Connectivity constraints for avoiding subtours.

3. Constraints ensuring that exactly one node from each cluster must be chosen

4. Degree constraints of the nodes (the degree of each node must be either zero or two).

In the following we use the notation $a(C^k)$, $\forall C^k \in 2^C$ to determine all inter-cluster edges of the clusters in C^k. Then, a valid lower bound on the partial solution $(x_m, \ldots, x_{m-l+1})$ can be computed by summing up costs of five different components:

1. The dynamic programming algorithm for the solution evaluation is adapted to work correctly for partial solutions and is executed on the fixed set of clusters, which results in the value lb^f. Note that lb^f already contains an arc from C_0 to C_l, although C_l is definitely not the first visited cluster, so the arc weight $a^f = \max_{j \in C_l} c_0 j$ is subtracted from lb^f resulting in a lower bound on C^f of $lb_1 = lb^f - a^f$.

2. The total cost of the $|C^o| - 1$ cheapest edges in C^o is denoted by lb_2. While there are methods that can produce better bounds, e.g., computing a minimum spanning tree, we choose to use this simple computation to keep the time consumption low.

3. A lower bound on the restocking costs for the clusters in C^o is computed by first taking the total sum of the expected demand $E[C^o]$ of these clusters. Then, lb_3 is given by multiplying the cheapest edge from the depot to any node in C^o, $\lfloor \frac{E[C^o]}{Q} \rfloor$ times.

4. To connect C^o to C^f the cheapest edge from C^o to C_l determines lb_4.

5. Finally, lb_5 is given by the cheapest edge from C_0 to any node in C^o.
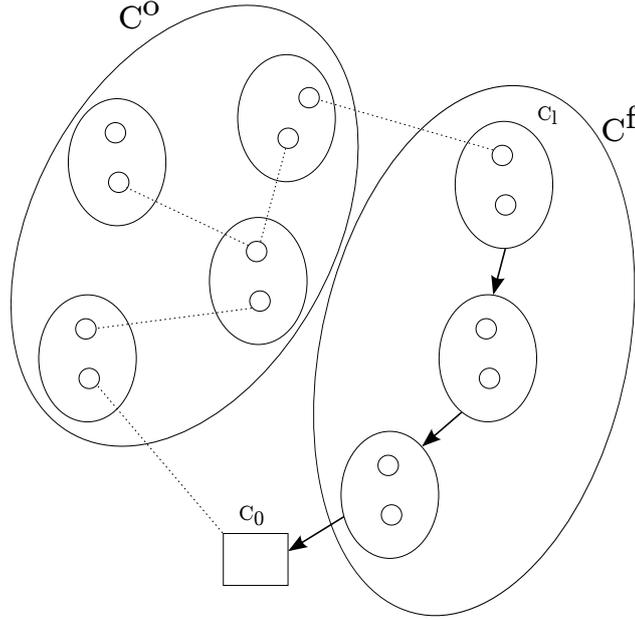
Figure 4 **Example of the computation of a lower bound on a partial solution.**

These individual parts form the lower bound $lb = \sum_{i=1}^{5} lb_i$, which is stored at the corresponding trie node. Directly after this computation or whenever this trie node is visited again, this lower bound is compared to the value of the best solution found so far, which corresponds to a global upper bound, to possibly cut off this node and the corresponding solution subspace. Figure 4 shows an example of the bounding procedure, where the position of three clusters are already fixed as denoted by the arrows. The dotted lines represent the lowest-cost edges which form lb_2, lb_4, and lb_5.

To speed up the computation of lb_1 at the cost of potentially worsening the bound, any approximation level from the multi-level evaluation (see Section 4.1) can be chosen for the DP algorithm. In our preliminary tests it turned out that even at the highest approximation level $\lceil \log_2 Q \rceil$ the bound was reasonably good so that this level is chosen for the remaining computational tests. However, even with this speed-up computing bounds on each new trie node would be too time consuming and therefore this procedure is only applied with a certain probability whenever a trie node is accessed.

5. Computational Results

To evaluate the developed algorithm a computational study is performed. We rely on a set of 158 benchmark instances for the GVRPSD (https://www.ac.tuwien.ac.at/research/problem-instances/#Generalized_Vehicle_Routing_Problem_with_Stochastic_Demands), which is also used by Biesinger et al. (2015c) and Biesinger et al. (2015b). These instances are based on (deterministic) instances for the generalized vehicle routing problem generated by Bektaş et al. (2011). They modified instances from the CVRP-library (http://branchandcut.org/VRP/data/), having 16 to 262 nodes and partitioned them into $m = \lfloor \frac{n}{\theta} \rfloor$ clusters, where $\theta = \{2, 3\}$. Biesinger et al.

(2015c,b) adapted these instances to the GVRPSD by setting the expected demand of each cluster to the deterministic demand of the corresponding cluster. Then the clusters are divided into *low spread* and *high spread* clusters uniformly at random. The possible demand values for each cluster lie in $\pm 10\%$ of the expected demand for *low spread* clusters and $\pm 30\%$ for *high spread* clusters. Values lower than zero or larger than Q are not considered. A uniform distribution is used for the set of possible demands. The algorithm is implemented in C++ using CPLEX 12.6 for solving the GTSP in the initial solution creation phase. All runs were executed on a single core of an Intel Xeon processor with 2.54 GHz and 20 GB RAM.

Each run of all tested configurations was repeated 30 times and terminated after a maximum of 300 CPU seconds ($T_{\max}=300$). Preliminary tests showed that the parameters for the basic GA were not particularly sensitive to changes, therefore the population size is fixed to 100, p_{mut} to 0.1, and *nMut* to 10.

In the first set of experiments the VND is examined more closely to evaluate the effectiveness of the used neighborhood structures. Then, extensive tests regarding the solution archive are performed. Therefore, the algorithm is run with and without the solution archive and results are compared. After that, the bounding extension is investigated in detail.

5.1.   Variable Neighborhood Descent

In a first step tests with various values for α, which determines the frequency of VND executions, are performed. In preliminary tests it turned out that when α is higher than 0.1 the run-time spent in the VND dominates the other parts of the algorithm too much. Therefore, we consider here $\alpha \in \{0.01, 0.05, 0.1\}$. Table 1 shows a summary of the results grouped by the instance set. In this table as well as in other tables in this chapter $\overline{obj}$ stands for the average objective value over 30 independent runs using all instances of the respective group, $\overline{obj_g}$ is the geometric mean over these runs, and $\overline{gap}$ *to BKS* is the average percentage gap to the best known solution (BKS). The BKS is determined by taking the best objective value of each instance separately over all runs and configurations which are tested here. The row #*Best results* indicates the number of instances, for which this configuration yields the best average objective value of the configurations under comparison. The next three rows show the p-Values of one-sided paired Wilcoxon rank sum tests which were performed over all instances.

The results in Table 1 show that $\overline{obj}$, $\overline{obj_g}$, and the gap to the BKS for $\alpha = 0.05$ and $\alpha = 0.1$ are lower than for $\alpha = 0.01$. The conclusion that the configuration with $\alpha = 0.01$ is worse than the other values for α is also confirmed by the statistical tests, which showed that both $\alpha = 0.05$ and $\alpha = 0.1$ are significantly better with an error level of less than 1%. Considering the number of best results both configurations with $\alpha \in \{0.05, 0.1\}$ have similar values and the gap to the BKS is lower for $\alpha = 0.05$

Table 1   Performance of the GA+VND with different values for α.

	Instances with $\theta = 2$			Instances with $\theta = 3$		
	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.1$
$\overline{obj}$	696.32	695.53	695.23	492.59	490.74	490.74
$\overline{obj_g}$	540.19	539.58	539.75	399.48	399.19	398.97
$\overline{gap}$ to BKS	1.31%	1.19%	1.25%	0.77%	0.70%	0.64%
# Best results	61	65	72	67	74	83
p-Value ($< \alpha = 0.01$)	-	0.000011	<0.000001	-	0.000002	<0.000001
p-Value ($< \alpha = 0.05$)	0.999989	-	0.006113	0.999998	-	0.082960
p-Value ($< \alpha = 0.1$)	>0.999999	0.993892	-	>0.999999	0.917141	-

when instances with $\theta = 2$ are considered but higher for the instance set with $\theta = 3$. However, the statistical test revealed that $\alpha = 0.1$ is significantly better for instances with $\theta = 2$ and therefore this value is used for the remaining tests.

To investigate the effectiveness of each of the used neighborhood structures within the VND the number of times where an improvement could be achieved was counted for each neighborhood $\mathcal{N}_1$ to $\mathcal{N}_4$. For this test we include the solution archive as it is the basis for $\mathcal{N}_4$. The number of conversions, which corresponds to the size of this neighborhood structure, is set to n^2 to have a comparable size to the other neighborhoods. Table 2 shows the number of improvements for each instance, which are referred to by their name, the number of nodes n, the number of clusters m, and the expected number of restocks $E[\mathrm{nr}]$.

As we see in Table 2 the number of improvements achieved in a specific neighborhood structure is usually smaller than the number in the preceding neighborhood structure. This can be explained by the design of the employed VND: each time an improvement in any neighborhood is found, the search restarts with the first neighborhood structure, so the earlier neighborhood structures are searched more often. But still it can be observed that for most but the largest instances even $\mathcal{N}_4$ was useful as it sometimes led to an improvement where the other neighborhood structures did not. Note that the numbers for the smallest instances are very small compared to the larger instances. This is because by using the solution archive the optimal solution was found within seconds after which the algorithm terminated. In Section 5.3 this issue is further discussed.

5.2. Solution Archive

The next experiments are performed to investigate the impact of the solution archive to the overall results. Therefore, the algorithm with the parameters described before (*without SA*) is compared to the algorithm with the solution archive and no conversion within the VND (*GASA $SA_{\mathrm{conv}}=0$*) and to the configuration with solution archive and conversion (*GASA $SA_{\mathrm{conv}}=1$*). Furthermore, the relative number of identified duplicate solutions per instance (dups) is recorded.

The results of these tests are shown in Table 3 for instances with $\theta = 2$ and in Table 4 for instances with $\theta = 3$. The average objective values ($\overline{obj}$) and corresponding standard deviations (sd) over 30

Table 2   Average number of improvements in the different neighborhood structures of the VND.

$\theta = 2$	n	m	$E[\text{nr}]$	$\mathcal{N}_1$	$\mathcal{N}_2$	$\mathcal{N}_3$	$\mathcal{N}_4$	$\theta = 3$	n	m	$E[\text{nr}]$	$\mathcal{N}_1$	$\mathcal{N}_2$	$\mathcal{N}_3$	$\mathcal{N}_4$
P1	16	8	4.20	38	8	5	48	P1	16	6	3.00	3	0	0	1
P2	19	10	1.08	10	13	1	2	P2	19	7	0.71	0	0	0	0
P3	20	10	1.04	28	16	3	6	P3	20	7	0.68	0	0	0	0
P4	21	11	1.04	268	110	26	46	P4	21	7	0.64	0	0	0	0
P5	22	11	1.04	147	47	14	21	P5	22	8	0.73	0	0	0	0
P6	22	11	3.87	380	166	110	40	P6	22	8	2.82	6	3	1	2
P7	23	12	4.53	11907	2694	2339	2652	P7	23	8	2.55	34	13	6	41
B1	31	16	2.03	8779	2648	953	951	B1	31	11	1.38	8701	3000	1034	3189
A1	32	16	2.00	10100	2741	706	887	A1	32	11	1.39	117	47	14	32
A2	33	17	2.33	9333	3272	722	736	A2	33	11	1.52	43	25	2	9
A3	33	17	2.74	9436	4039	818	658	A3	33	11	1.91	98	39	9	15
A4	34	17	2.25	9099	3345	778	666	A4	34	12	1.66	793	265	79	122
B2	34	17	2.05	7243	2395	521	791	B2	34	12	1.34	748	423	59	151
B3	35	18	2.23	7421	1946	505	621	B3	35	12	1.54	4596	1630	432	973
A5	36	18	1.98	6572	2860	484	494	A5	36	12	1.34	771	223	52	125
A6	37	19	2.10	2552	1127	151	163	A6	37	13	1.43	147	57	6	16
A7	37	19	2.93	6754	2520	575	571	A7	37	13	1.95	6282	1949	719	906
A8	38	19	2.42	7011	2513	518	430	A8	38	13	1.71	257	133	25	33
B4	38	19	2.75	5291	1989	447	758	B4	38	13	1.93	8167	2799	702	1708
A9	39	20	2.47	6163	1995	494	350	A9	39	13	1.48	1478	449	101	184
A10	39	20	2.68	6002	1971	491	456	A10	39	13	1.83	1743	897	185	335
B5	39	20	2.33	5753	1660	153	508	B5	39	13	1.45	3	4	0	0
P8	40	20	2.26	5956	2021	325	284	P8	40	14	1.51	215	118	13	29
B6	41	21	2.88	5344	1922	243	339	B6	41	14	1.82	4751	1654	210	584
B7	43	22	2.78	5037	1359	350	296	B7	43	15	1.81	2355	1171	238	265
A11	44	22	2.91	4818	1526	371	367	A11	44	15	2.00	6241	1921	508	802
B8	44	22	3.16	4030	1255	303	332	B8	44	15	2.24	5579	1726	361	785
A12	45	23	3.12	4234	1528	335	275	A12	45	15	2.09	6596	2174	439	732
A13	45	23	3.16	3749	1265	401	328	A13	45	15	2.06	5538	1682	450	860
B9	45	23	2.45	5486	1344	213	211	B9	45	15	1.51	602	200	51	76
B10	45	23	3.05	3865	1361	367	338	B10	45	15	1.96	6417	2155	353	673
P9	45	23	2.35	3529	1123	190	220	P9	45	15	1.61	1202	457	73	140
A14	46	23	3.12	3856	1379	309	270	A14	46	16	2.08	5478	1816	409	635
A15	48	24	3.25	3474	1070	329	331	A15	48	16	2.13	4531	1188	352	709
B11	50	25	3.01	3260	1223	155	283	B11	50	17	2.05	3539	1195	121	301
B12	50	25	4.26	2336	282	210	274	B12	50	17	2.72	4624	1508	385	716
P10	50	25	4.77	3364	994	298	280	P10	50	17	3.32	4934	1437	336	725
P11	50	25	3.18	2952	803	169	208	P11	50	17	2.21	3689	1264	171	452
P12	50	25	3.98	3059	966	203	217	P12	50	17	2.77	4266	1164	239	586
B13	51	26	3.56	2693	866	234	239	B13	51	17	2.44	5710	1249	282	454
P13	51	26	5.04	3123	862	275	237	P13	51	17	3.31	4964	1255	350	617
B14	52	26	3.12	2799	872	161	224	B14	52	18	2.18	3622	1412	266	474
A16	53	27	3.38	2688	687	197	164	A16	53	18	2.09	4178	1176	253	348
A17	54	27	3.43	2477	812	226	227	A17	54	18	2.19	3966	939	220	356
A18	55	28	4.18	2344	639	189	147	A18	55	19	2.75	3818	1362	294	353
P14	55	28	4.72	2244	702	186	169	P14	55	19	3.21	3353	830	215	400
P15	55	28	7.76	2094	257	258	253	P15	55	19	5.27	3580	402	344	763
P16	55	28	3.19	2189	654	127	115	P16	55	19	2.17	2983	859	134	253
P17	55	28	3.39	2260	661	122	118	P17	55	19	2.31	3041	771	149	269
B15	56	28	3.19	1753	413	91	111	B15	56	19	2.20	4013	1134	123	279
B16	57	29	3.68	2394	543	119	107	B16	57	19	2.39	2724	598	104	188
B17	57	29	4.19	1831	511	144	161	B17	57	19	2.55	2928	840	256	445
A19	60	30	4.11	1700	589	173	135	A19	60	20	2.80	3010	875	229	330
P18	60	30	4.83	1748	521	143	126	P18	60	20	3.19	2728	694	179	331
P19	60	30	7.23	1684	530	195	165	P19	60	20	4.78	3306	871	284	474
A20	61	31	4.51	1528	423	143	115	A20	61	21	3.09	2801	897	202	322
A21	62	31	3.66	1605	504	127	78	A21	62	21	2.59	2586	577	185	300
A22	63	32	4.69	1372	420	137	92	A22	63	21	3.14	2493	826	219	308
A23	63	32	4.52	1384	477	135	119	A23	63	21	2.96	2437	723	205	344
B18	63	32	4.41	1589	422	129	107	B18	63	21	2.90	2611	740	184	363
A24	64	32	4.11	1342	357	138	107	A24	64	22	2.55	2409	592	169	276
B19	64	32	4.17	1361	427	105	114	B19	64	22	3.17	2951	701	159	256
A25	65	33	4.14	1092	397	122	89	A25	65	22	2.89	2472	701	166	266
P20	65	33	4.78	1189	375	116	90	P20	65	22	3.23	2592	562	143	241
B20	66	33	4.36	1458	376	93	85	B20	66	22	2.80	2427	625	156	251
B21	67	34	4.59	1425	298	90	71	B21	67	23	3.14	2217	563	153	240
B22	68	34	4.17	1303	383	81	77	B22	68	23	2.87	2257	552	121	232
A26	69	35	4.30	940	302	89	69	A26	69	23	2.94	2340	542	116	196
P21	70	35	4.79	946	299	82	64	P21	70	24	3.36	2062	478	111	206
P22	76	38	1.95	529	172	12	12	P22	76	26	1.33	659	210	25	30
P23	76	38	2.44	697	204	29	18	P23	76	26	1.67	966	237	26	39
B23	78	39	4.89	1056	183	45	37	B23	78	26	3.31	1798	358	79	132
A27	80	40	4.43	627	150	56	37	A27	80	27	3.04	1388	267	83	116
M1	101	51	4.73	197	38	7	6	M1	101	34	3.20	1071	81	11	16
P24	101	51	1.83	179	33	4	1	P24	101	34	1.25	318	50	6	12
M2	121	61	3.54	113	14	1	1	M2	121	41	2.44	226	52	11	12
M3	151	76	5.69	87	3	0	0	M3	151	51	3.71	119	19	3	3
M4	200	100	7.92	46	0	0	0	M4	200	67	5.29	63	0	0	0
G1	262	131	11.79	7	0	0	0	G1	262	88	8.13	9	0	0	0

Table 3   Results for the configurations without the solution archive (without SA), with the SA and no conversion in the VND (GASA $SA_{conv}=0$), and with the SA and with conversion (GASA $SA_{conv}=1$) for instances with $\theta = 2$

$\theta = 2$	without SA		GASA $SA_{conv}=0$			GASA $SA_{conv}=1$		
	$\overline{obj}$	sd	$\overline{obj}$	sd	dups[%]	$\overline{obj}$	sd	dups[%]
P1	245.34	0.00	245.34	0.00	39.89	245.34	0.00	33.48
P2	146.82	0.00	146.82	0.00	93.01	146.82	0.00	92.86
P3	149.02	0.00	149.02	0.00	92.64	149.02	0.00	92.92
P4	160.48	0.00	160.48	0.00	86.81	160.48	0.00	85.24
P5	162.95	0.00	**161.36**	0.00	86.75	**161.36**	0.00	87.48
P6	323.59	0.01	323.59	0.00	24.28	323.59	0.01	21.64
P7	312.51	0.00	312.51	0.01	17.76	312.51	0.01	15.81
B1	419.91	0.00	419.91	0.00	11.95	419.91	0.00	11.06
A1	538.49	3.49	**520.04**	0.00	41.98	**520.40**	0.94	24.93
A2	455.15	0.00	455.15	0.00	39.80	455.15	0.00	26.17
A3	468.10	0.19	**467.95**	0.00	45.02	**467.96**	0.07	23.23
A4	503.34	2.07	**498.15**	0.00	46.05	**498.35**	1.09	26.59
B2	466.80	0.00	466.80	0.00	13.40	466.80	0.00	12.53
B3	619.24	0.00	619.24	0.00	12.89	619.24	0.00	11.52
A5	506.95	0.00	**506.40**	0.79	58.80	**506.68**	0.63	40.58
A6	454.29	3.95	**447.86**	0.00	76.72	**447.86**	0.00	57.30
A7	609.17	3.78	**590.59**	3.49	15.18	**595.03**	8.16	11.05
A8	481.97	0.00	481.97	0.00	29.78	481.97	0.00	20.95
B4	479.92	0.00	479.92	0.00	10.18	479.87	0.26	9.50
A9	567.41	0.00	567.41	0.00	16.84	567.41	0.00	11.75
A10	561.25	0.00	**560.61**	0.40	15.91	**561.02**	0.39	11.52
B5	356.43	0.00	356.43	0.00	16.33	356.43	0.00	16.44
P8	296.44	0.00	**296.36**	0.05	17.50	**296.35**	0.04	18.46
B6	483.26	0.00	483.22	0.15	9.58	483.26	0.00	9.08
B7	485.46	0.00	485.46	0.00	9.22	485.46	0.00	8.61
A11	627.86	0.00	627.86	0.00	10.66	627.86	0.00	8.93
B8	563.95	0.00	563.95	0.00	8.19	563.95	0.00	7.81
A12	621.23	0.00	621.23	0.00	10.29	621.23	0.00	8.44
A13	692.89	0.00	692.89	0.00	8.24	693.09	1.09	7.47
B9	502.02	0.00	502.02	0.00	10.76	502.02	0.00	10.52
B10	482.91	0.00	482.91	0.00	7.95	482.91	0.00	7.44
P9	340.68	0.00	**340.50**	0.06	10.16	**340.53**	0.09	9.60
A14	624.05	0.00	**622.84**	1.32	8.03	**622.75**	1.32	7.54
A15	686.42	0.00	686.42	0.00	7.62	686.42	0.00	7.25
B11	454.09	0.00	454.09	0.00	7.92	454.09	0.00	7.68
B12	923.53	0.00	923.53	0.00	6.81	923.53	0.00	6.49
P10	423.34	0.57	**422.24**	1.50	7.06	**421.86**	1.45	6.58
P11	354.47	0.00	354.47	0.00	8.51	354.47	0.00	8.12
P12	377.66	0.00	377.62	0.21	7.39	377.62	0.21	6.97
B13	682.70	0.00	682.70	0.00	6.85	682.70	0.00	6.47
P13	451.79	0.00	451.79	0.00	6.86	451.79	0.00	6.51
B14	458.66	0.33	**458.39**	0.34	7.00	**458.34**	0.33	6.65
A16	633.47	0.00	632.78	2.79	6.55	**632.03**	4.48	6.21
A17	722.02	2.62	721.54	3.63	6.50	722.38	0.60	6.12
A18	718.26	0.51	718.11	0.06	6.33	718.45	1.80	6.03
P14	420.69	0.00	420.69	0.00	6.30	420.69	0.00	5.97
P15	560.92	0.01	560.86	0.30	6.21	**560.69**	0.60	5.93
P16	361.87	0.00	361.87	0.00	6.84	361.87	0.00	6.53
P17	362.07	0.03	**362.04**	0.02	7.03	**362.03**	0.00	6.70
B15	474.92	0.00	474.92	0.00	6.09	474.89	0.15	5.84
B16	779.30	0.41	**778.60**	0.91	5.94	**778.76**	0.92	5.69
B17	967.33	0.00	967.33	0.00	5.91	967.33	0.00	5.72
A19	815.86	0.00	815.86	0.00	5.79	815.86	0.00	5.59
P18	452.86	0.00	452.86	0.00	5.78	452.86	0.00	5.45
P19	572.08	0.00	572.08	0.00	5.85	572.06	0.37	5.50
A20	658.06	8.24	653.64	9.18	5.60	**653.30**	8.81	5.46
A21	755.75	0.00	755.75	0.00	5.81	755.75	0.00	5.51
A22	830.88	0.53	830.88	0.53	5.45	830.80	0.66	5.25
A23	946.39	0.00	946.39	0.00	5.44	946.39	0.00	5.22
B18	852.87	0.00	852.87	0.00	5.54	852.87	0.00	5.28
A24	837.31	0.00	837.31	0.00	5.41	837.31	0.00	5.12
B19	514.92	0.00	514.92	0.00	5.32	514.92	0.00	5.05
A25	712.14	0.00	712.14	0.00	5.42	712.14	0.00	5.11
P20	501.39	0.00	501.34	0.30	5.43	501.39	0.00	5.25
B20	818.42	0.00	818.42	0.00	5.15	818.42	0.00	4.92
B21	672.40	0.00	672.40	0.00	4.93	672.42	0.05	4.80
B22	738.48	0.00	738.48	0.00	4.97	738.13	1.91	4.76
A26	706.39	6.91	707.90	6.07	4.85	707.67	6.52	4.68
P21	504.96	0.00	504.96	0.00	4.90	504.96	0.00	4.69
P22	394.20	0.00	**392.81**	1.01	8.55	**392.89**	0.89	8.43
P23	409.93	0.00	409.93	0.00	6.08	409.93	0.00	5.96
B23	873.27	12.83	863.99	18.93	4.00	865.28	18.52	3.80
A27	1049.26	0.71	1049.26	0.71	3.79	1049.39	0.00	3.60
M1	569.06	17.30	569.15	19.41	3.47	572.23	18.08	3.44
P24	462.25	3.27	**458.43**	0.35	28.98	**458.37**	0.28	24.97
M2	896.42	83.21	860.22	112.30	3.09	886.63	148.42	2.88
M3	966.97	102.47	983.18	209.26	2.39	993.70	211.74	2.20
M4	1967.18	166.82	1680.24	605.77	2.02	1740.09	545.62	1.71
G1	8709.61	775.64	8669.10	741.93	1.52	8526.76	550.58	1.21

Table 4 **Results for the configurations without the solution archive (without SA), with the SA and no conversion in the VND (GASA $SA_{\mathrm{conv}}=0$), and with the SA and with conversion (GASA $SA_{\mathrm{conv}}=1$) for instances with $\theta = 3$**

$\theta = 3$	without SA		GASA $SA_{\mathrm{conv}}=0$			GASA $SA_{\mathrm{conv}}=1$		
	$\overline{obj}$	sd	$\overline{obj}$	sd	dups[%]	$\overline{obj}$	sd	dups[%]
P1	170.37	0.00	170.37	0.00	7.46	170.37	0.00	10.36
P2	112.10	0.00	112.10	0.00	70.94	112.10	0.00	72.02
P3	117.31	0.00	117.31	0.00	72.52	117.31	0.00	72.00
P4	117.07	0.00	117.07	0.00	71.52	117.07	0.00	70.90
P5	111.19	0.00	111.19	0.00	87.77	111.19	0.00	87.66
P6	245.83	0.00	245.83	0.00	31.92	245.83	0.00	29.28
P7	183.59	0.00	183.59	0.00	20.44	183.59	0.00	19.78
B1	355.73	0.00	355.73	0.00	28.09	355.73	0.00	24.59
A1	386.91	0.00	386.91	0.00	85.45	386.91	0.00	85.33
A2	318.03	0.00	318.03	0.00	89.28	318.03	0.00	88.76
A3	364.59	0.00	364.59	0.00	86.83	364.59	0.00	87.40
A4	419.12	0.00	419.12	0.00	71.34	419.12	0.00	70.76
B2	363.09	0.00	363.09	0.00	72.98	363.09	0.00	73.14
B3	501.47	0.00	**500.87**	0.00	45.85	**500.87**	0.00	47.17
A5	399.90	0.00	399.90	0.00	74.41	399.90	0.00	71.99
A6	359.13	0.00	359.13	0.00	83.34	359.13	0.00	83.65
A7	430.99	0.00	430.99	0.00	48.11	430.99	0.00	48.19
A8	371.80	0.00	371.80	0.00	80.60	371.80	0.00	78.99
B4	389.04	0.77	**386.25**	0.30	28.90	**386.25**	0.30	24.60
A9	371.41	0.00	371.41	0.00	63.80	371.41	0.00	64.09
A10	416.03	0.00	416.03	0.00	59.30	416.03	0.00	60.26
B5	281.48	0.00	281.48	0.00	88.47	281.48	0.00	88.64
P8	214.75	0.00	214.75	0.00	80.90	214.75	0.00	78.42
B6	404.26	0.00	404.26	0.00	39.34	404.26	0.00	39.92
B7	347.65	0.00	347.65	0.00	57.08	347.65	0.00	55.01
A11	508.85	0.70	**506.60**	1.09	26.19	**506.48**	1.03	24.82
B8	402.02	0.00	402.02	0.00	14.85	402.02	0.00	14.13
A12	478.22	0.00	478.22	0.00	26.14	478.22	0.00	24.67
A13	488.02	0.00	488.02	0.00	11.66	488.02	0.00	11.53
B9	417.03	0.00	417.03	0.00	66.42	417.03	0.00	67.98
B10	358.99	0.00	358.99	0.00	16.81	358.99	0.00	15.99
P9	239.36	0.00	239.36	0.00	59.32	239.36	0.00	58.08
A14	470.96	0.00	**466.82**	2.22	18.53	**466.34**	1.85	18.58
A15	462.55	0.00	462.55	0.00	15.58	462.55	0.00	14.26
B11	398.38	0.00	398.38	0.00	15.97	398.38	0.00	14.11
B12	600.66	0.00	600.64	0.05	7.87	600.65	0.03	7.63
P10	302.37	0.00	302.37	0.00	8.00	302.37	0.00	8.30
P11	261.31	0.00	261.31	0.00	21.17	261.31	0.00	17.35
P12	273.12	0.80	**268.91**	0.00	18.91	**268.91**	0.00	16.08
B13	513.02	0.00	513.02	0.00	12.84	513.02	0.00	12.34
P13	313.41	0.00	313.41	0.00	7.57	313.41	0.00	7.87
B14	360.50	0.00	360.50	0.00	22.16	360.50	0.00	18.56
A16	443.87	0.00	443.87	0.00	17.87	443.87	0.00	17.86
A17	490.54	0.00	490.54	0.00	10.00	490.54	0.00	9.86
A18	474.05	0.00	474.05	0.00	10.24	474.05	0.00	9.24
P14	316.65	0.00	**313.37**	2.18	6.99	**313.67**	2.25	6.72
P15	396.20	0.00	396.20	0.00	6.38	396.12	0.44	6.10
P16	274.22	0.00	274.22	0.00	12.14	274.22	0.00	12.97
P17	276.33	0.00	276.33	0.00	10.92	276.33	0.00	11.55
B15	358.81	0.26	**357.91**	0.26	10.65	**357.84**	0.00	10.44
B16	565.26	1.06	**564.53**	0.55	9.42	**564.37**	0.12	9.09
B17	689.78	8.96	**681.73**	11.45	6.35	682.01	11.30	6.14
A19	620.10	3.03	616.92	5.72	6.67	616.68	8.13	6.28
P18	328.89	0.00	**328.83**	0.05	6.41	**328.83**	0.05	5.94
P19	372.63	0.00	372.63	0.00	5.83	372.63	0.00	5.66
A20	482.51	0.00	482.51	0.00	6.11	482.51	0.00	5.96
A21	617.56	0.00	617.56	0.00	6.31	617.56	0.00	6.02
A22	611.54	0.00	611.54	0.00	5.86	611.66	0.68	5.69
A23	665.59	1.46	664.95	1.64	5.84	664.66	1.61	5.67
B18	604.67	0.07	604.66	0.09	6.80	604.62	0.10	6.53
A24	564.46	0.00	564.46	0.00	6.44	564.33	0.73	6.25
B19	457.24	0.00	457.24	0.00	8.13	457.24	0.00	7.62
A25	525.03	0.00	525.03	0.00	6.06	525.03	0.00	5.78
P20	378.48	0.00	378.48	0.00	5.68	378.48	0.00	5.56
B20	627.36	0.00	627.22	0.35	7.29	627.29	0.26	7.14
B21	561.71	0.00	561.71	0.00	5.74	561.71	0.00	5.58
B22	538.59	1.91	539.25	1.46	6.01	538.70	1.89	5.74
A26	523.77	0.00	523.77	0.00	7.94	523.77	0.00	7.84
P21	386.15	0.00	**385.82**	0.19	5.28	**385.88**	0.21	4.93
P22	310.40	0.00	310.40	0.00	34.38	310.40	0.00	32.96
P23	310.40	0.00	310.40	0.00	23.82	310.40	0.00	27.64
B23	620.11	0.00	620.11	0.00	7.73	620.11	0.00	7.48
A27	748.99	6.03	752.06	6.18	5.25	752.06	6.15	5.09
M1	468.10	14.60	463.48	8.51	5.32	463.06	5.95	5.33
P24	378.68	0.00	**371.93**	0.00	18.22	**372.74**	2.15	18.26
M2	578.24	18.02	**561.00**	18.40	4.87	566.74	18.97	4.57
M3	538.05	30.39	532.71	44.21	4.13	523.75	33.84	4.04
M4	936.83	113.82	**834.52**	158.87	2.89	**829.13**	163.78	2.59
G1	3769.61	317.90	3843.03	324.70	2.31	3898.56	302.18	2.09

runs are given along with the number of duplicate solutions relative to all generated solutions. The bold numbers in the column for the configuration without the SA mean that on these instances the algorithm without the SA achieved statistically better results (using a pairwise Wilcoxon rank sum test as described before) than either GASA SA_{conv}=0 or GASA SA_{conv}=1. The bold numbers in the other columns indicate statistically better results for the respective configuration compared only to the GA without the SA. These tables show that through all instance sizes the SA is able to improve the algorithm as it produces in 20 out of 79 instances with $\theta = 2$ and in 14 out of 79 instances with $\theta = 3$ significantly better results whereas the GA without the archive was never significantly better. When considering the number of duplicate solutions, it is observed that generally the larger the instances the fewer duplicates are produced and the average number of duplicates is 16.30% for instances with $\theta = 2$ and 28.29% for instances with $\theta = 3$ (for SA_{conv}=0). A summary of the results is given in Table 5 where it becomes more obvious that both configurations using the SA achieve significantly better results than the GA without the SA at a significance level of 1%. However, when comparing SA_{conv}=0 to SA_{conv}=1 the results do not show a clear indication which one performed better, but SA_{conv}=0 is used for the remaining tests since the total average gap to the BKS and the geometric mean is lower for this configuration.

Table 5   **Performance of the GA with different variants of the SA.**

	Instances with $\theta = 2$			Instances with $\theta = 3$		
	no SA	SA_{conv}=0	SA_{conv}=1	no SA	SA_{conv}=0	SA_{conv}=1
$\overline{obj}$	679.91	674.57	674.12	460.92	459.77	460.36
$\overline{obj_g}$	545.15	542.90	543.41	398.58	397.42	397.42
$\overline{gap}$ to BKS	3.22%	2.58%	2.72%	1.37%	1.02%	1.02%
# Best results	43	60	55	56	62	65
p-Value ($< $ no SA)	-	<0.000001	<0.000001	-	<0.000001	<0.000001
p-Value ($< SA_{\mathrm{conv}}$=0)	>0.999999	-	0.949769	>0.999999	-	0.641107
p-Value ($< SA_{\mathrm{conv}}$=1)	>0.999999	0.050273	-	>0.999999	0.359077	-

5.3. Bounding Extension

For the evaluation of the bounding extension the probability of a bound computation on the visit of a trie node is set to 5% and as already stated in Section 4.4.3 the coarsest approximation of the DP is used for computing lb_1.

To investigate the impact of the bounding extension on the algorithm first it is determined how successful the bound computations are. In this context *successful* means that after the bound computation the subtrie could actually be cut off, i.e., the computed lower bound on this partial solution is already higher than the global upper bound given by the best solution found so far. To get an overview of the results the relative number of bound cuts is grouped by the trie levels which are divided into four quarters in which the bounds are computed.

Table 6 **Successful bound cuts grouped by instance and part of the trie where they were computed.**

$\theta = 2$	0-25[%]	26-50[%]	51-75[%]	76-100[%]	$\theta = 3$	0-25[%]	26-50[%]	51-75[%]	76-100[%]
P1	0.00	0.00	0.00	0.00	P1	0.00	0.00	0.00	0.00
P2	24.36	68.98	94.08	98.62	P2	23.10	85.17	97.84	100.00
P3	16.35	58.85	91.73	98.70	P3	7.04	55.04	96.43	100.00
P4	8.64	56.54	91.24	98.87	P4	2.11	56.65	98.20	99.77
P5	8.64	61.21	93.34	99.26	P5	26.20	69.65	95.93	98.44
P6	0.00	5.97	36.61	58.74	P6	0.66	10.95	38.90	52.22
P7	0.00	0.00	0.60	1.48	P7	0.00	1.51	12.33	22.65
B1	2.04	5.70	19.93	52.47	B1	0.55	5.82	42.14	75.16
A1	18.51	29.46	55.61	81.62	A1	16.36	61.68	92.43	99.06
A2	13.59	26.76	57.28	82.25	A2	10.27	63.27	94.08	99.53
A3	11.83	28.28	57.13	82.00	A3	6.73	65.43	92.35	99.12
A4	20.76	35.39	60.46	84.44	A4	29.18	52.41	80.32	95.01
B2	3.34	13.65	49.10	77.29	B2	27.58	61.50	88.67	96.21
B3	0.46	9.85	42.46	76.86	B3	22.92	44.71	75.78	89.48
A5	19.48	35.85	64.10	88.70	A5	10.22	52.40	84.36	96.48
A6	23.82	48.40	68.69	92.24	A6	16.09	55.16	85.11	96.95
A7	1.58	9.65	39.31	77.19	A7	14.90	36.45	72.13	87.82
A8	9.48	27.05	48.70	83.96	A8	33.88	62.06	85.18	96.72
B4	0.33	8.27	35.36	63.84	B4	21.58	34.48	68.48	84.00
A9	1.69	11.78	38.91	73.40	A9	11.22	41.05	77.66	93.15
A10	2.77	12.80	40.60	74.55	A10	17.42	44.38	77.05	91.96
B5	14.93	27.88	58.20	82.91	B5	59.44	82.71	97.39	99.78
P8	9.79	21.41	45.14	77.91	P8	23.69	53.42	74.92	94.77
B6	1.42	7.33	44.00	80.95	B6	29.05	45.74	74.50	91.72
B7	0.03	3.35	32.67	73.67	B7	31.74	57.65	77.59	94.08
A11	0.53	6.60	36.27	73.91	A11	2.17	24.03	54.29	84.30
B8	0.01	1.31	24.68	64.77	B8	1.60	15.38	52.98	80.71
A12	1.60	9.32	39.19	76.30	A12	3.32	21.33	47.93	81.74
A13	0.02	1.37	20.29	60.32	A13	0.32	6.15	36.58	73.74
B9	2.24	11.99	46.06	85.50	B9	53.73	63.08	81.45	96.77
B10	0.38	3.58	22.70	66.08	B10	3.68	15.53	47.88	81.54
P9	5.20	17.22	41.87	81.12	P9	14.92	42.35	59.49	90.87
A14	0.38	5.55	33.19	72.75	A14	6.72	19.84	46.65	76.94
A15	0.12	2.80	21.94	59.50	A15	3.73	14.90	44.38	75.62
B11	2.54	10.94	45.15	77.34	B11	10.57	19.04	57.26	84.89
B12	0.00	0.02	5.66	37.12	B12	0.00	1.24	29.63	60.06
P10	0.06	1.74	24.28	61.04	P10	0.32	3.76	33.82	65.34
P11	1.73	7.19	35.85	73.10	P11	7.13	19.16	46.49	76.01
P12	0.55	5.01	33.22	68.40	P12	5.18	16.18	45.38	74.69
B13	0.20	4.94	32.80	66.08	B13	8.01	24.95	56.29	82.17
P13	0.03	1.56	21.64	60.19	P13	0.11	2.22	28.23	61.43
B14	0.68	7.17	40.21	74.88	B14	14.54	27.65	60.58	87.43
A16	0.42	4.92	30.92	71.31	A16	7.38	18.42	49.76	82.64
A17	0.03	1.57	20.36	64.63	A17	1.94	10.90	41.67	76.49
A18	0.25	3.45	25.48	64.52	A18	4.70	15.58	44.44	79.71
P14	0.15	2.83	21.75	59.34	P14	0.10	3.02	25.61	67.13
P15	0.00	0.16	7.35	39.64	P15	0.00	0.24	13.44	49.72
P16	1.45	8.56	32.95	69.56	P16	2.40	11.29	35.00	75.81
P17	1.75	8.84	33.94	70.90	P17	1.98	10.80	35.30	75.28
B15	0.61	5.57	31.63	66.08	B15	4.68	15.89	48.02	84.96
B16	0.02	0.91	19.40	60.83	B16	0.35	7.42	39.14	79.63
B17	0.00	0.09	7.49	36.96	B17	0.01	1.92	17.44	59.09
A19	0.01	0.92	16.79	58.12	A19	0.13	3.15	26.76	64.46
P18	0.10	2.77	24.13	62.89	P18	0.31	3.47	25.64	61.94
P19	0.00	0.34	10.13	47.70	P19	0.01	1.32	18.72	51.65
A20	0.17	2.92	23.89	64.39	A20	0.43	4.09	32.11	68.87
A21	0.13	2.75	23.80	68.27	A21	0.15	3.76	29.69	67.98
A22	0.02	1.33	16.55	54.39	A22	0.10	2.50	30.90	66.13
A23	0.00	0.62	10.36	45.26	A23	0.04	1.52	20.93	57.48
B18	0.03	1.99	16.62	53.05	B18	1.45	10.06	38.99	73.59
A24	0.02	1.13	15.13	53.13	A24	0.38	5.70	31.05	70.06
B19	0.86	6.54	24.01	60.17	B19	0.68	6.38	34.56	74.20
A25	0.34	4.48	28.94	63.92	A25	1.04	6.89	34.28	71.82
P20	0.10	1.81	22.24	61.10	P20	0.11	2.12	23.11	63.25
B20	0.00	0.71	13.62	51.56	B20	0.18	5.68	31.98	72.01
B21	0.07	2.27	19.61	58.78	B21	0.20	3.65	30.50	72.79
B22	0.01	0.80	14.97	54.15	B22	0.21	4.30	30.09	73.62
A26	0.15	2.65	23.24	65.45	A26	0.93	6.18	30.37	74.23
P21	0.07	1.38	17.69	60.77	P21	0.22	2.21	20.72	60.28
P22	4.94	11.84	38.76	79.72	P22	18.93	28.92	47.04	80.43
P23	1.86	8.18	34.45	77.95	P23	15.82	24.08	45.51	79.60
B23	0.01	0.37	9.88	51.07	B23	0.14	2.87	28.00	70.48
A27	0.01	0.55	9.56	45.63	A27	0.03	1.25	13.42	59.07
M1	0.41	2.51	14.68	45.68	M1	0.30	2.35	19.23	64.11
P24	9.74	14.03	36.32	80.61	P24	7.23	14.18	35.56	75.67
M2	0.12	0.34	12.33	46.12	M2	0.55	1.70	24.17	63.63
M3	0.20	0.61	8.73	33.08	M3	0.43	1.22	12.03	52.82
M4	0.00	0.00	0.00	17.81	M4	0.04	0.05	0.84	27.47
G1	0.00	0.00	0.00	8.43	G1	0.00	0.00	0.00	13.10

Table 6 shows the number of successful bound cuts. Column 0-25 corresponds to the top quarter of the trie, column 26-50 to the second quarter and so on. As expected, this value increases on higher levels as more of the solution is already fixed. However, also for the lower levels this number is surprisingly high for some of the smaller instances. Even if the number of bound cuts on the first quarter of the trie is often less than 1% for the larger instances, one successful cut on a top level drastically reduces the search space, as a cut on level i removes $(m-i+1)!$ solution candidates, which are not considered in later iterations anymore. On average the successful bound cuts for instances with $\theta = 2$ are, starting from the first quarter, 3.22%, 10.42%, 31.75%, and 64.47% and for the instances with $\theta = 3$, 8.25%, 22.55%, 47.79%, 75.31%.

5.4. Optimal Solutions

Using the solution archive and the bounding extension within the GA has the side effect to enhance the algorithm into an exact bounded enumeration method. This is basically a theoretic result but if the computed bounds are strong enough it may be sufficient to solve smaller instances to proven optimality. Therefore, a set of experiments is conducted in which the global run-time was not limited but instead a memory limit of 20 GB is used.

Table 7   List of the instances which could be solved optimally within the memory limit of 20 GB with and without the bounding extension of the solution archive.

$\theta = 2$	obj*	GASA t[s]	GASA +bounding t[s]	$\theta = 3$	obj*	GASA t[s]	GASA +bounding t[s]
P1	245.34	1.16	0.86	P1	170.37	0.06	0.06
P2	146.82	63.56	22.30	P2	112.10	0.11	0.08
P3	149.02	63.34	31.48	P3	117.31	0.13	0.11
P4	160.48	756.62	282.02	P4	117.07	0.13	0.11
P5	161.36	756.54	300.48	P5	111.19	0.72	0.46
P6	323.59	1445.44	1335.61	P6	245.83	6.31	7.20
P7	312.48	13334.70	14126.00	P7	183.59	0.98	0.79
				B1	355.73	782.63	721.70
				A1	386.91	835.08	288.87
				A2	318.03	842.92	332.92
				A3	364.59	832.20	306.27
				A4	419.12	10697.70	2538.82
				B2	363.09	10595.70	1959.18
				B3	500.87	10740.90	3144.64
				A5	399.90	10702.10	3802.57
				A8	371.80	-	13415.50
				B5	281.48	-	6535.06

Table 7 shows the instances, along with their optimal objective values and the needed time, which could be solved within the memory limit with and without the bounding extension. We observe that all instances with up to 12 clusters could be solved optimally. The largest instance which could be solved was B5 with $\theta = 3$, 39 nodes, and 13 clusters, which is solved in less than two hours. For most of the solved instances the bounding extension is able to reduce the needed run-time by up to a

factor of more than 5.0 (for instance B2) and for the largest two instances it is even able to find the optimal solution whereas the GA with the solution archive alone could not.

5.5. Final Results

Finally, the GASA and the GASA with the bounding extension are compared to each other and to the only already existing heuristic algorithm for the GVRPSD, the VNS by Biesinger et al. (2015c). All algorithms are terminated after 300 seconds and in Table 8 the average objective values over 30 runs and the corresponding standard deviations are shown. As in Section 5.2 bold values indicate that the corresponding algorithm performed significantly better than the other two methods on a 1% error level according to a one-sided paired Wilcoxon rank sum test.

The results show that both GASA and GASA + bounding outperform the VNS on most of the instances. Specifically, GASA found on 35 instances with $\theta = 2$ and on 28 instances with $\theta = 3$ significantly better results and GASA + bounding on 38 instances with $\theta = 2$ and on 30 instances with $\theta = 3$ better results than the VNS. The VNS, however, achieved only in five instances better results than any of the other two algorithms. When we compare GASA with GASA + bounding in Table 8 it is not clear which should be preferred. Therefore, an overall summary of all three algorithms is given in Table 9, which is constructed like the previous summaries. Although GASA + bounding has a higher arithmetic mean than GASA, the geometric mean, the average gap to the BKS, and the number of best results are better which is also reflected in the statistical tests which showed that GASA + bounding performs significantly better on the given problem instances. Additionally it has the property that for smaller instances it can actually find proven optimal solutions which makes GASA + bounding the superior algorithm.

6. Conclusions and Outlook

In this work an evolutionary algorithm using a complete trie-based solution archive was developed to solve the generalized vehicle routing problem with stochastic demand using the (optimal) preventive restocking strategy. The genetic algorithm was enhanced by a variable neighborhood search using four neighborhood structures in order to intensify the search. The solution archive was enhanced by a bounding extension such that the considered solution space could be significantly pruned. All components were analyzed to show their individual contribution to the performance of the overall algorithm.

Extensive computational experiments were conducted to analyze the effectiveness of the presented algorithm. The results show the superiority of the GASA method compared to an existing VNS. By the fast calculation of lower bounds on partial solutions the search space can be significantly reduced which lead to even better solutions on some instances. Furthermore, by using the solution archive

Table 8   Results of the GASA with and without the bounding extension and the VNS from the literature.

	VNS		GASA		GASA + bounding			VNS		GASA		GASA + bounding	
$\theta = 2$	$\overline{obj}$	sd	$\overline{obj}$	sd	$\overline{obj}$	sd	$\theta = 3$	$\overline{obj}$	sd	$\overline{obj}$	sd	$\overline{obj}$	sd
P1	245.34	0.00	245.34	0.00	245.34	0.00	P1	170.37	0.00	170.37	0.00	170.37	0.00
P2	146.82	0.00	146.82	0.00	146.82	0.00	P2	112.10	0.00	112.10	0.00	112.10	0.00
P3	149.02	0.00	149.02	0.00	149.02	0.00	P3	117.31	0.00	117.31	0.00	117.31	0.00
P4	160.48	0.00	160.48	0.00	160.48	0.00	P4	117.07	0.00	117.07	0.00	117.07	0.00
P5	161.36	0.00	161.36	0.00	161.36	0.00	P5	111.19	0.00	111.19	0.00	111.19	0.00
P6	323.95	0.91	**323.59**	0.00	**323.59**	0.00	P6	245.83	0.00	245.83	0.00	245.83	0.00
P7	312.51	0.00	312.51	0.01	312.51	0.01	P7	183.59	0.00	183.59	0.00	183.59	0.00
B1	419.91	0.00	419.91	0.00	419.91	0.00	B1	355.73	0.00	355.73	0.00	355.73	0.00
A1	521.92	5.73	520.04	0.00	520.04	0.00	A1	386.91	0.00	386.91	0.00	386.91	0.00
A2	455.34	0.25	**455.15**	0.00	**455.15**	0.00	A2	318.03	0.00	318.03	0.00	318.03	0.00
A3	468.76	0.08	**467.95**	0.00	**467.95**	0.00	A3	364.59	0.00	364.59	0.00	364.59	0.00
A4	498.15	0.00	498.15	0.00	498.15	0.00	A4	419.12	0.00	419.12	0.00	419.12	0.00
B2	466.80	0.00	466.80	0.00	466.80	0.00	B2	363.09	0.00	363.09	0.00	363.09	0.00
B3	619.24	0.00	619.24	0.00	619.24	0.00	B3	501.39	0.21	**500.87**	0.00	**500.87**	0.00
A5	506.95	0.00	**506.40**	0.79	**506.46**	0.77	A5	399.90	0.00	399.90	0.00	399.90	0.00
A6	447.86	0.00	447.86	0.00	447.86	0.00	A6	359.13	0.00	359.13	0.00	359.13	0.00
A7	608.39	0.86	590.59	3.49	**589.70**	0.71	A7	430.99	0.00	430.99	0.00	430.99	0.00
A8	481.98	0.00	**481.97**	0.00	**481.97**	0.00	A8	371.80	0.00	371.80	0.00	371.80	0.00
B4	**479.44**	0.69	479.92	0.00	479.92	0.00	B4	388.84	1.05	**386.25**	0.30	**386.25**	0.30
A9	567.91	0.00	**567.41**	0.00	**567.41**	0.00	A9	371.41	0.00	371.41	0.00	371.41	0.00
A10	561.25	0.00	**560.61**	0.40	**560.73**	0.44	A10	417.78	0.33	**416.03**	0.00	**416.03**	0.00
B5	356.48	0.00	**356.43**	0.00	**356.43**	0.00	B5	281.48	0.00	281.48	0.00	281.48	0.00
P8	296.44	0.00	296.36	0.05	**296.33**	0.00	P8	214.75	0.00	214.75	0.00	214.75	0.00
B6	483.26	0.00	483.22	0.15	483.20	0.18	B6	404.26	0.00	404.26	0.00	404.26	0.00
B7	487.02	2.20	**485.46**	0.00	**485.46**	0.00	B7	347.65	0.00	347.65	0.00	347.65	0.00
A11	627.86	0.00	627.86	0.00	627.86	0.00	A11	508.98	0.00	**506.60**	1.09	**505.32**	0.59
B8	563.96	0.00	**563.95**	0.00	**563.95**	0.00	B8	402.02	0.00	402.02	0.00	402.02	0.00
A12	621.23	0.00	621.23	0.00	621.23	0.00	A12	478.22	0.00	478.22	0.00	478.22	0.00
A13	692.89	0.00	692.89	0.00	692.89	0.00	A13	488.02	0.00	488.02	0.00	488.02	0.00
B9	502.02	0.00	502.02	0.00	502.02	0.00	B9	419.35	0.79	**417.03**	0.00	**417.03**	0.00
B10	482.91	0.00	**482.91**	0.00	**482.91**	0.00	B10	358.99	0.00	358.99	0.00	358.99	0.00
P9	340.48	0.00	340.50	0.06	340.49	0.04	P9	239.36	0.00	239.36	0.00	239.36	0.00
A14	623.01	1.16	622.84	1.32	622.58	1.31	A14	471.34	0.50	466.82	2.22	**465.62**	0.00
A15	686.42	0.00	686.42	0.00	686.42	0.00	A15	462.55	0.00	462.55	0.00	462.55	0.00
B11	454.09	0.00	454.09	0.00	454.09	0.00	B11	398.38	0.00	398.38	0.00	398.38	0.00
B12	923.53	0.00	923.53	0.00	923.53	0.00	B12	604.92	1.64	**600.64**	0.05	**600.62**	0.06
P10	431.22	1.31	**422.24**	1.50	**421.36**	1.33	P10	302.37	0.00	302.37	0.00	302.37	0.00
P11	354.47	0.00	354.47	0.00	354.47	0.00	P11	261.31	0.00	**261.31**	0.00	**261.31**	0.00
P12	377.66	0.00	377.62	0.21	377.62	0.19	P12	273.27	0.00	**268.91**	0.00	**268.91**	0.00
B13	682.27	1.32	**682.70**	0.00	**682.70**	0.00	B13	513.02	0.00	**513.02**	0.00	**513.02**	0.00
P13	451.79	0.00	451.79	0.00	451.64	0.52	P13	313.41	0.00	313.41	0.00	313.41	0.00
B14	458.87	0.14	**458.39**	0.34	**458.57**	0.19	B14	360.50	0.00	360.50	0.00	360.50	0.00
A16	636.61	1.89	632.78	2.79	631.10	5.99	A16	443.87	0.00	443.87	0.00	443.87	0.00
A17	721.48	2.64	721.54	3.63	720.96	4.12	A17	490.54	0.00	490.54	0.00	490.54	0.00
A18	730.53	4.91	**718.11**	0.06	**718.12**	0.00	A18	474.05	0.00	474.05	0.00	474.05	0.00
P14	424.54	0.19	**420.69**	0.00	**420.69**	0.00	P14	316.65	0.00	313.37	2.18	**312.14**	0.87
P15	560.92	0.00	560.86	0.30	560.62	0.67	P15	395.57	0.76	396.20	0.00	396.16	0.27
P16	370.43	5.70	**361.87**	0.00	**361.87**	0.00	P16	274.22	0.00	274.22	0.00	274.22	0.00
P17	362.21	0.00	**362.04**	0.02	**362.03**	0.00	P17	276.33	0.00	**276.33**	0.00	**276.33**	0.00
B15	478.10	0.00	**474.92**	0.00	**474.92**	0.00	B15	358.85	0.19	357.91	0.26	**357.84**	0.00
B16	779.43	0.19	**778.60**	0.91	**778.69**	1.13	B16	567.66	0.23	564.53	0.55	**564.35**	0.00
B17	967.33	0.00	967.33	0.00	967.33	0.00	B17	692.38	3.10	**681.73**	11.45	**674.93**	8.63
A19	816.39	0.00	**815.86**	0.00	**815.86**	0.00	A19	617.87	4.70	616.92	5.72	615.61	5.95
P18	455.26	0.00	**452.86**	0.00	**452.86**	0.00	P18	328.89	0.00	328.83	0.05	328.79	0.00
P19	572.08	0.00	572.08	0.00	572.07	0.09	P19	372.63	0.00	372.63	0.00	372.63	0.00
A20	662.94	0.00	653.64	9.18	**648.92**	7.85	A20	482.51	0.00	482.51	0.00	482.51	0.00
A21	755.77	0.00	**755.75**	0.00	**755.75**	0.00	A21	617.56	0.00	617.56	0.00	617.56	0.00
A22	**830.79**	0.00	830.88	0.53	830.88	0.53	A22	611.54	0.00	611.54	0.00	611.54	0.00
A23	946.39	0.00	946.39	0.00	946.39	0.00	A23	666.46	0.00	**664.95**	1.64	**663.65**	1.12
B18	852.87	0.00	852.87	0.00	852.87	0.00	B18	604.68	0.00	604.66	0.09	604.59	0.11
A24	837.31	0.00	837.31	0.00	837.31	0.00	A24	563.57	3.39	564.46	0.00	564.02	2.44
B19	514.92	0.00	514.92	0.00	514.92	0.00	B19	457.24	0.00	**457.24**	0.00	**457.24**	0.00
A25	712.74	0.00	**712.14**	0.00	**712.14**	0.00	A25	525.03	0.00	525.03	0.00	525.03	0.00
P20	501.39	0.00	501.34	0.30	501.39	0.00	P20	378.53	0.00	**378.48**	0.00	**378.48**	0.00
B20	818.42	0.00	818.42	0.00	818.42	0.00	B20	627.36	0.00	627.22	0.35	627.22	0.35
B21	674.95	0.00	**672.40**	0.00	**672.40**	0.00	B21	561.71	0.00	561.71	0.00	561.71	0.00
B22	738.48	0.00	738.48	0.00	738.48	0.00	B22	539.10	1.63	539.25	1.46	538.88	1.74
A26	711.19	0.00	707.90	6.07	708.78	5.48	A26	523.77	0.00	523.77	0.00	523.77	0.00
P21	504.96	0.00	504.96	0.00	504.96	0.00	P21	386.07	0.17	**385.82**	0.19	**385.69**	0.00
P22	394.16	0.20	**392.81**	1.01	**392.46**	1.12	P22	310.40	0.00	310.40	0.00	310.40	0.00
P23	409.93	0.00	409.93	0.00	409.93	0.00	P23	310.40	0.00	**310.40**	0.00	**310.40**	0.00
B23	**840.94**	0.52	863.99	18.93	**839.53**	0.00	B23	620.11	0.00	**620.11**	0.00	**620.11**	0.00
A27	1064.86	0.00	**1049.26**	0.71	**1049.13**	0.98	A27	757.24	1.70	**751.46**	6.18	**743.00**	0.00
M1	590.38	0.03	**569.15**	19.41	**544.82**	0.55	M1	467.14	0.03	**463.48**	8.51	**463.96**	0.00
P24	462.18	0.83	**458.43**	0.35	**458.31**	0.23	P24	371.93	0.00	371.93	0.00	371.93	0.00
M2	**769.86**	0.00	860.22	112.30	**745.93**	0.01	M2	565.77	0.00	**561.00**	18.40	**545.87**	1.25
M3	**732.85**	0.80	983.18	209.26	**692.56**	0.41	M3	530.05	3.73	532.71	44.21	**506.51**	20.49
M4	3372.85	97.94	**1680.24**	605.77	**1398.44**	69.30	M4	2338.33	45.86	**834.52**	158.87	**646.41**	17.22
G1	13817.9	252.8	**8669.10**	741.93	**10741.71**	338.27	G1	8544.93	161.56	**3843.03**	324.70	**4435.65**	345.10

Table 9 **Summary of the performance of the GASA with and without the bounding extension compared to the VNS.**

	Instances with $\theta = 2$			Instances with $\theta = 3$		
	VNS	GASA	GASA+bounding	VNS	GASA	GASA+bounding
$\overline{obj}$	758.17	674.57	691.39	539.00	459.77	464.08
$\overline{obj_g}$	549.48	542.90	539.14	407.29	397.42	396.28
$\overline{gap}$ to BKS	5.66%	2.58%	1.70%	6.14%	1.02%	0.72%
# Best results	31	53	66	48	56	74
p-Value (< VNS)	-	<0.000001	<0.000001	-	<0.000001	<0.000001
p-Value (< GASA)	>0.999999	-	<0.000001	>0.999999	-	0.000948
p-Value (< GASA+bounding)	>0.999999	>0.999999	-	>0.999999	0.999053	-

and the bounding extension the algorithm is able to find proven optimal solutions for 24 instances with up to 39 nodes and 13 clusters.

Ideas for future work include the application of this method to similar problems, e.g., when a maximum route duration is given such that more than one tour has to be planned. Also a more in-depth analysis when the GASA is applied to the VRPSD could be interesting. For the solution archive another promising research direction is the utilization of the computed bounds for making a more intelligent branching decision.

Acknowledgments

References

Afsar, H Murat, Christian Prins, Andréa Cynthia Santos. 2014. Exact and heuristic algorithms for solving the generalized vehicle routing problem with flexible fleet size. *International Transactions in Operational Research* **21**(1) 153–175.

Bektaş, Tolga, Güneş Erdoğan, Stefan Røpke. 2011. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science* **45**(3) 299–316.

Bertsimas, Dimitris J. 1992. A vehicle routing problem with stochastic demand. *Operations Research* **40**(3) 574–585.

Bianchi, Leonora, Mauro Birattari, Marco Chiarandini, Max Manfrin, Monaldo Mastrolilli, Luis Paquete, Olivia Rossi-Doria, Tommaso Schiavinotto. 2006. Hybrid metaheuristics for the vehicle routing problem with stochastic demands. *Journal of Mathematical Modelling and Algorithms* **5**(1) 91–110.

Biesinger, Benjamin, Bin Hu, Günther Raidl. 2014a. Models and algorithms for competitive facility location problems with different customer behavior. *Annals of Mathematics and Artificial Intelligence* 1–27doi: 10.1007/s10472-014-9448-0.

Biesinger, Benjamin, Bin Hu, Günther Raidl. 2015a. A hybrid genetic algorithm with solution archive for the discrete $(r|p)$-centroid problem. *Journal of Heuristics* **21**(3) 391–431.

Biesinger, Benjamin, Bin Hu, Günther R. Raidl. 2014b. An evolutionary algorithm for the leader-follower facility location problem with proportional customer behavior. *Conference Proceedings of Learning and Intelligent Optimization Conference (LION 8), LNCS*, vol. 8426. Springer, 203–217.

Biesinger, Benjamin, Bin Hu, Günther R. Raidl. 2015b. An integer L-shaped method for the generalized vehicle routing problem with stochastic demands. *International Network Optimization Converence (INOC 2015)*. Springer, 1–8. To appear.

Biesinger, Benjamin, Bin Hu, Günther R. Raidl. 2015c. A variable neighborhood search for the generalized vehicle routing problem with stochastic demands. Gabriela Ochoa, Francisco Chicano, eds., *Evolutionary Computation in Combinatorial Optimization – EvoCOP 2015*, *LNCS*, vol. 9026. Springer, 48–60.

Christiansen, Christian H, Jens Lysgaard. 2007. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters* **35**(6) 773–781.

Fischetti, Matteo, Juan José Salazar González, Paolo Toth. 1997. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research* **45**(3) 378–394.

Gendreau, Michel, Gilbert Laporte, René Séguin. 1995. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation science* **29**(2) 143–155.

Gendreau, Michel, Gilbert Laporte, René Séguin. 1996. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research* **44**(3) 469–477.

Ghiani, Gianpaolo, Gennaro Improta. 2000. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research* **122**(1) 11–17.

Goodson, Justin C, Jeffrey W Ohlmann, Barrett W Thomas. 2012. Cyclic-order neighborhoods with application to the vehicle routing problem with stochastic demand. *European Journal of Operational Research* **217**(2) 312–323.

Hà, Minh Hoàng, Nathalie Bostel, André Langevin, Louis-Martin Rousseau. 2014. An exact algorithm and a metaheuristic for the generalized vehicle routing problem with flexible fleet size. *Computers & Operations Research* **43** 9 – 19.

Hansen, Pierre, Nenad Mladenović. 2001. Variable neighborhood search: Principles and applications. *European Journal of Operational Research* **130**(3) 449 – 467.

Hjorring, Curt, John Holt. 1999. New optimality cuts for a single-vehicle stochastic routing problem. *Annals of Operations Research* **86** 569–584.

Hu, Bin, Günther R. Raidl. 2012a. An evolutionary algorithm with solution archive for the generalized minimum spanning tree problem. R. Moreno-Díaz, F. Pichler, A. Quesada-Arencibia, eds., *Proceedings of the 13th International Conference on Computer Aided Systems Theory: Part I*, *LNCS*, vol. 6927. Springer, 287–294.

Hu, Bin, Günther R. Raidl. 2012b. An evolutionary algorithm with solution archives and bounding extension for the generalized minimum spanning tree problem. *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (GECCO)*. ACM Press, Philadelphia, PA, USA, 393–400.

Jabali, Ola, Walter Rei, Michel Gendreau, Gilbert Laporte. 2014. Partial-route inequalities for the multi-vehicle routing problem with stochastic demands. *Discrete Applied Mathematics* **177** 121–136.

Kara, Imdat, Tolga Bektaş. 2003. Integer linear programming formulation of the generalized vehicle routing problem. *Proc. of the 5-th EURO/INFORMS Joint International Meeting*. 06–10.

Laporte, Gilbert, François V Louveaux. 1998. Solving stochastic routing problems with the integer L-shaped method. TeodorGabriel Crainic, Gilbert Laporte, eds., *Fleet Management and Logistics*. Springer, 159–167.

Laporte, Gilbert, François V Louveaux, Luc Van Hamme. 2002. An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research* **50**(3) 415–423.

Marinaki, Magdalene, Yannis Marinakis. 2015. A glowworm swarm optimization algorithm for the vehicle routing problem with stochastic demands. *Expert Systems with Applications* .

Marinakis, Yannis, Georgia-Roumbini Iordanidou, Magdalene Marinaki. 2013. Particle swarm optimization for the vehicle routing problem with stochastic demands. *Applied Soft Computing* **13**(4) 1693–1704.

Marinakis, Yannis, Magdalene Marinaki. 2013. Combinatorial expanding neighborhood topology particle swarm optimization for the vehicle routing problem with stochastic demands. *Proceedings of the 15th annual Conference on Genetic and Evolutionary Computation*. ACM, 49–56.

Marinakis, Yannis, Magdalene Marinaki, Athanasios Migdalas. 2014. A hybrid clonal selection algorithm for the vehicle routing problem with stochastic demands. *Learning and Intelligent Optimization*. Springer, 258–273.

Marinakis, Yannis, Magdalene Marinaki, Paraskevi Spanou. 2015. A memetic differential evolution algorithm for the vehicle routing problem with stochastic demands. Iztok Fister, Iztok Fister Jr., eds., *Adaptation and Hybridization in Computational Intelligence*, *Adaptation, Learning, and Optimization*, vol. 18. Springer, 185–204.

Mauldin, M L. 1984. Maintaining Diversity in Genetic Search. *National Conference on Artificial Intelligence*, vol. 19. AAAI, William Kaufmann, 247–250.

Pop, Petrica Claudiu, Levente Fuksz, Andrei Horvat Marc. 2014. A variable neighborhood search approach for solving the generalized vehicle routing problem. Marios Polycarpou, AndréC.P.L.F. de Carvalho, Jeng-Shyang Pan, Michałł Woźniak, Héctor Quintian, Emilio Corchado, eds., *Hybrid Artificial Intelligence Systems*, *LNCS*, vol. 8480. Springer, 13–24.

Pop, Petrica Claudiu, O. Matei, C.Pop Sitar, C. Chira. 2010. A genetic algorithm for solving the generalized vehicle routing problem. Emilio Corchado, Manuel Graña Romay, Alexandre Manhaes Savio, eds., *Hybrid Artificial Intelligence Systems*, *LNCS*, vol. 6077. Springer, 119–126.

Raidl, Günther, Bin Hu. 2010. Enhancing Genetic Algorithms by a Trie-Based Complete Solution Archive. Peter Cowling, Peter Merz, ed., *Evolutionary Computation in Combinatorial Optimisation – EvoCOP 2010*, *LNCS*, vol. 6022. Springer, 239–251.

Rei, Walter, Michel Gendreau, Patrick Soriano. 2010. A hybrid monte carlo local branching algorithm for the single vehicle routing problem with stochastic demands. *Transportation Science* **44**(1) 136–146.

Ronald, Simon. 1998. Duplicate Genotypes in a Genetic Algorithm. *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. 793–798.

Ruthmair, Mario, Günther R. Raidl. 2012. A memetic algorithm and a solution archive for the rooted delay-constrained minimum spanning tree problem. R. Moreno-Díaz, et al., eds., *Proceedings of the 13th International Conference on Computer Aided Systems Theory: Part I*, *LNCS*, vol. 6927. Springer, 351–358.

Stephen, Graham A. 1994. *String searching algorithms*. World Scientific.

Yang, Wen-Huei, Kamlesh Mathur, Ronald H. Ballou. 2000. Stochastic vehicle routing problem with restocking. *Transportation Science* **34**(1) 99–112.