Solving large-scale vehicle routing problem instances using an island-model offspring selection genetic algorithm

Stefan Vonolfen*, Michael Affenzeller†, Andreas Beham‡, Stefan Wagner§
School of Informatics, Communications and Media
Upper Austria University of Applied Sciences
Softwarepark 11
4232 Hagenberg
Austria
*stefan.vonolfen@heuristiclab.com
†michael.affenzeller@heuristiclab.com
‡andreas.beham@heuristiclab.com
§stefan.wagner@heuristiclab.com

Abstract—The vehicle routing problem is a class of problems that frequently occurs in the field of transportation logistics. In this work, we tackle very-large scale problem instances with time windows. Among other techniques, metaheuristics are frequently used to solve large-scale instances close to optimality. We present an island-model genetic algorithm variant and apply several techniques such as offspring selection and adaptive constraint relaxation. To validate our approach, we perform test runs on benchmark instances with 1000 customers and compare the results to the currently best-known solutions.

Index Terms—Vehicle routing problem, island-model genetic algorithm, offspring selection

I. INTRODUCTION

The vehicle routing problem (VRP) is an important problem class in the field of transport logistics optimization. The original formulation of the problem has been defined over 50 years ago by [1] and consists of a fleet of vehicles serving a set of customers with a certain demand from a single depot and the vehicles have a certain capacity. This formulation is referred to as the capacitated vehicle problem (CVRP).

In practice often variants with many diverse constraints occur. Those extended problem formulations are often called rich vehicle routing problems [2]. Thus, since then many diverse variants of vehicle routing problems have been studied in the literature, for a taxonomic review see for example [3]. The classification includes different scenario characteristics such as time windows, different problem physical characteristics such as the geographical location of customers and different information and data characteristics.

One important variant frequently studied in the literature is the capacitated vehicle routing problem with time windows (CVRPTW [4]). In addition to satisfying the capacity constraints, also time window constraints have to be considered. This means, that each customer has to be served within a certain time window and late or early arrivals are not allowed. Each customer requires a certain service time.

Similar to the diversity of the studied problem variants, there are also various solution methods proposed in the literature. Since the vehicle routing problem is known to be NP-hard [5], it becomes increasingly intractable in larger problem dimensions. Today, some problem instances of the CVRPTW with 100 customers are not solved to optimality because of the high complexity. Thus, metaheuristics are frequently used to generate feasible and near-optimal solutions for large scale instances. For an overview of different metaheuristics for the VRP see for example [6] or [7].

An overview of the state-of-the-art of solving large-scale VRPTW instances is given by [8] where they compare different algorithms in terms of their performance on large-scale problem instances. In terms of the VRPTW, problem instances up to 1000 customers can be solved efficiently using heuristics. Often hybrid variants are used to efficiently solve large-scale instances. For example [9] apply a hybridization of an evolution strategy with guided local search. In [10] a two-stage hybrid local search is used which consists a combination of large neighborhood search and simulated annealing. A hybridization of branch-and-price and large neighborhood search was developed by [11].

Among other metaheuristics, such as tabu search (TS) or variable neighborhood search (VNS), genetic algorithms (GA) have been used successfully to tackle large problem instances. In this work, we present a genetic algorithm variant that is capable of robustly solving different large-scale instances.

The rest of the paper is organized as following: in Section II we present our algorithmic approach, in Section III we perform test runs on several benchmark instances and in Section IV we summarize our findings and show how the approach could be extended in the future.
II. METHODOLOGY

In GA terminology, an individual denotes a solution to the VRP that is encoded in a certain representation. The representation (genotype) can be mapped to routes that can be carried out by the vehicles (phenotype). A genotype can be evaluated by simulating the tours while considering the capacity and time window constraints. The quality of an individual is then determined by the driven distance and the number of required vehicles. The decision process what tours should be carried out is very complex, especially in large problem dimensions.

To solve very-large scale CVRPTW instances we develop a special island-model GA variant that is capable of robustly delivering satisfying solution qualities in reasonable computation time without having to perform instance-specific tuning.

To achieve that the main success criteria are:

- Utilization of parallel computation power
- Interplay between breadth and depth search
- Preservation of essential genetic information
- Achieving progress in different stages of the search process
- Efficiently moving through the search space of feasible and infeasible solutions

We decided to apply and island-model GA because they are known to support the interplay of breadth and depth search and in that way support the efficient exploration and exploitation of the search space. In the case of combinatorial optimization problems (such as the VRP) [12] observe that the interplay between migration and the individual development of islands is beneficial for multimodal search spaces. Additionally the combinatorial complexity increases exponentially with the problem instance size. To solve the problem efficiently it is thus important to utilize the parallel computation power of modern computers. Table I shows the parameters of the island-model GA. The parameters have been tuned in several experiments and proved to be robust on several problem instances. The initial population is created using the push-forward insertion heuristic [13].

First empirical experiments showed that, especially in advanced stages of the search process, it is difficult for the operators to actually achieve improvement. Thus we decided to use a steady-state selection scheme which replaces a single individual with a successful one in each generation. As shown in several studies, such as [14], genetic drift is accelerated when using steady-state replacement instead of generational replacement. Additionally a relatively low migration interval and a relatively high mutation rate are used to keep the search process vital. These techniques aim at preventing premature convergence of the populations as a whole. The combination of fast convergence of individual islands in combination with frequent exchange of genetic information steers the interplay between depth and breadth search with the focus rather on the depth search component.

The reason why individually developing islands and coarse-grained migration did not work that well in our initial experiments may be due to the fact that it is hard to combine individual solutions that stem from different parts of the search space. This might be due to the fact that combining different individuals often leads to infeasible tours in terms of time windows. Thus, time windows make the search process even more intractable. This assumption is supported by the fact that CVRP instances with up to 24000 customers have been solved heuristically [15], whereas state-of-the-art algorithms for the CVRPTW are usually tested on instances with up to 1000 customers according to [8].

A. Offspring Selection

As stated earlier, in each generation for each island a single individual is created which replaces the current worst individual. To ensure the preservation of relevant genetic information during the search process, an offspring selection step is applied to the newly created individual. The concept, which has been developed by [16], ensures that the created individual is at least as good as its parents which are selected using random selection. This ensures that the operators actually produce beneficial offspring and no essential information is lost.

Table II lists the parameters of the offspring selection step. First two parents are selected randomly. Then, using crossover and mutation, a child is created. Whether the child is successful or not is determined by its fitness value $f_c$, the fitness values of the parents $f_{p1}, f_{p2}$ and by the comparison factor $c$. A child is considered successful, if it is at least as good as both parents:

$$f_c < \min(f_{p1}, f_{p2}) + (1-c)(\max(f_{p1}, f_{p2}) - \min(f_{p1}, f_{p2}))$$

If the child is successful, it is accepted and replaces the worst individual in the population. If it is rejected, the mating step is repeated until a successful child is created or the maximum selection pressure is reached. The selection pressure in that case is the count of the unsuccessfully created individuals. If no successful offspring could be created, a random unsuccessful one is taken.

<table>
<thead>
<tr>
<th>Population size</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of islands</td>
<td>8</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>25%</td>
</tr>
<tr>
<td>Migration rate</td>
<td>10%</td>
</tr>
<tr>
<td>Migration interval</td>
<td>5</td>
</tr>
<tr>
<td>Selection</td>
<td>Random</td>
</tr>
<tr>
<td>Replacement</td>
<td>Steady-state with offspring selection</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>PARAMETER SETTINGS FOR OFFSPRING SELECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparison factor</td>
<td>0.5</td>
</tr>
<tr>
<td>Maximum selection pressure</td>
<td>150</td>
</tr>
<tr>
<td>Success ratio</td>
<td>10%</td>
</tr>
</tbody>
</table>
B. Encodings and Operations

A crucial part for the success of a genetic algorithms are the problem-specific mutation and crossover operations that operate on a specific encoding. Unlike for other problems, such as the traveling salesman problem (TSP), no widely accepted default encoding for the VRP has been developed yet. In [12] several interesting encodings are identified which are tested and compared to each other by [17] where the conclusion is that it is beneficial to combine different encodings and operations in one algorithm run.

Another interesting observation was, that operators perform unequally well in different stages of the search process. Some operators yield large improvements at the beginning of the search process whereas some operators are beneficial when fine-tuning already good solutions. This is related to the fact that some operators are more disruptive than others.

The operations described by Potvin [18] operate on an encoding where the solutions are represented as a list of tours. The crossover and mutation operations are designed specifically for the CVRPTW. Experiments showed that they are often able to outperform the other operations [17]. Similar to that, the generic vehicle routing (GVR) concept proposed by Pereira [19] also encodes the tours directly and provides specific crossover and mutation operations, however in addition to that uses a repair function to avoid overload on the tours.

As an effort to utilize the well-known permutation encoding, the operations proposed by Prins [20] are based on a permutation encoding without trip delimiters in conjunction with standard permutation operations. It is based on a route-first, cluster second approach. The individual tours are determined using a specific split-procedure. A permutation encoding without trip delimiters is also used by Zhu [21]. Specific crossover operations are used in conjunction with standard permutation operations. In contrast to that, Alba [22] uses a permutation encoding with trip delimiters and thus encodes the individual tours directly and applies several well-known permutation operations.

The different encodings can be converted to each other, which means that if an operation is executed on an individual which is represented in a different encoding, the routes are extracted from that individual and then converted to the respective encoding. Each individual can be converted from its genotype to its phenotype and then be converted back to another genotype.

Because of the multitude of different crossover and manipulation operation it is beneficial to execute those with a higher probability that have a higher chance of producing successful offspring. In other words, the algorithm adaptively executes those crossover and manipulation operations proportional to their success in the current stage of the search process.

The relevant parameters for the adaptive parameter execution are listed in Table III. At each iteration $i$ the successfully created offspring is analyzed and for the crossover ($o_c$) and mutation ($o_m$) operator that was applied on it, the execution probability $p$ is increased by a constant factor $\alpha$:

$$p_{o_c}^{i+1} = p_{o_c}^i \cdot (1 + \alpha)$$

and

$$p_{o_m}^{i+1} = p_{o_m}^i \cdot (1 + \alpha)$$

After that step, the probabilities of all operators are normalized, so they sum up to 1. To prevent an operator to reach a probability close to zero, there is a lower bound (minimum probability). This ensures that operators that are not successful right now can still be executed later in the search process.

Figure 1 shows an example algorithm run and the development of the execution probabilities. Some operators are more successful in the beginning, some can produce better offspring towards the end of the search process.

C. Evaluation

In our case, the fitness of an individual is calculated the following way:

$$f(I) = \alpha \cdot d(I) + \beta \cdot r(I) + \gamma \cdot o(I) + \delta \cdot t(I)$$

where $f(I)$ is the fitness of the individual $I$, $d(I)$ the driven distance, $r(I)$ the count of the routes, $o(I)$ the overload violation and $t(I)$ the tardiness violation.

The parameters $\alpha$ and $\beta$ are set as fixed values and weight the importance of the distance in relation to the used fleet. To ensure the validity of the solutions found, soft constraints are used. Soft constraints generally have the advantage of

<table>
<thead>
<tr>
<th>Parameter Settings for the Adaptive Parameter Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update factor ($\alpha$)</td>
</tr>
<tr>
<td>Minimum probability</td>
</tr>
</tbody>
</table>

![Figure 1: Operator execution probabilities during the run of an algorithm](image)
creating a smoother search space. If hard constraints are used, regions of the search space are invalid and cannot be reached at all. When using soft constraints, the individual is penalized proportional to its constraint violation. The penalty is controlled by the parameters $\gamma$ and $\delta$. The penalty decreases the quality of an individual, since we consider a minimization problem.

To efficiently move through the search space of feasible and infeasible solutions, the penalty parameters are adapted during the algorithm run. At the beginning of the algorithm run they are initialized for each island with a value in the range of the interval $[10, 100]$.

Then, at each iteration $i$ they are adapted according to the following formula:

$$\gamma = \gamma / \phi + (\gamma * \sigma - \gamma / \phi) * \rho$$

and

$$\delta = \delta / \phi + (\delta * \sigma - \delta / \phi) * \rho$$

where $\sigma$ is the factor $> 1$ that increases the penalty and $\phi$ is the factor $> 1$ that decreases the penalty. The parameter $\rho$ denotes the percentage of infeasible solutions in the population and is in the interval $[0, 1]$. That means, that if all solutions in the island are infeasible, the penalty gets multiplied by $\sigma$. On the other hand, if all solutions are feasible, the penalty gets divided by $\phi$.

This self-adaptive steering of the penalties allows the steering of the search process in terms of how intensive infeasible parts of the search space are exploited to reach another feasible solution. This can be beneficial, if for example a crossover operator creates an initially infeasible solution that eventually leads to a better feasible solution.

The parameter setting of the evaluation is listed in Table IV. Our primary objective is to optimize the number of tours. $\phi$ was set higher than $\sigma$ which means that the search process should stay longer in valid regions than in invalid ones.

### Table IV

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>$\beta$</td>
<td>1300</td>
</tr>
<tr>
<td>$\phi$</td>
<td>1.04</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>1.01</td>
</tr>
</tbody>
</table>

We compared our results to the current best results available online\(^2\) (accessed on June 17th 2011). For the comparison of the total quality we applied our fitness function to the best known solution. In our evaluation we weighted the fleet usage with a factor of 1500 and the distance with a factor of 1.

The results are listed in Table V. Tests were carried out with a maximum of 10 million evaluated solutions. Each row in the table represents the aggregated results for each problem instance class, each includes 10 different instances. The results are compared to the best known results in terms of used vehicles, distance and the fitness function we used in our algorithm runs (quality).

The results show that the approach can deliver good results for all problem instance types without instance specific tuning. However, the results also indicate that even though the genetic algorithm variants includes several algorithmic extensions, there is still a gap to hybrid and two-stage algorithms that were used to generate the current best-known solutions. Also, problem specific tuning would be necessary to reach better results for the individual problem instances.

When compared to the currently best-known methods for solving very large scale VRP instances, our method can deliver good results in terms of the used vehicles. However, there is still a considerable gap when it comes to the minimization of the driven distance. We expect, that this gap could be closed by applying a two-stage approach which minimizes the driven distance separately in an additional step. In terms of runtime, many solution evaluations are required to reach good results. Here the hybridization with single-solution based techniques could lead to a considerable increase in runtime efficiency.

### IV. Conclusion and outlook

We have developed an island genetic algorithm variant for solving large-scale vehicle routing problem instances. To solve large problem instances efficiently we have incorporated offspring selection, an adaptive constraint relaxation technique and an adaptive execution of successful operations. The results show that the method can deliver good results for different instance classes of the Homberger and Gehring benchmark set with 1000 customers without problem specific tuning. However, there is still a gap compared to the currently best-known methods.

Thus, in the future it would be interesting to create hybrid variants and to incorporate local search concepts into the genetic algorithm. It is expected that the results can be improved while utilizing fewer evaluations using a hybrid variant. Also two-stage approaches are very promising. Additionally, the results should be compared to other parallel metaheuristics such as the SASEGASA [12].

---

1\(^{http://www.ferruni-hagen.de/WINF/touren/inhalte/proбинst.htm}\)

2\(^{http://www.sintef.no/Projectweb/TOP/Problems/VRPTW/Homberger-benchmark/1000-customers/}\)
### TABLE V

<table>
<thead>
<tr>
<th>Instances</th>
<th>Vehicles</th>
<th>Distance</th>
<th>Quality</th>
<th>Vehicles</th>
<th>Distance</th>
<th>Quality</th>
<th>Vehicles</th>
<th>Distance</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1_10</td>
<td>941</td>
<td>438612.83</td>
<td>1850112.83</td>
<td>964</td>
<td>440162.61</td>
<td>1886162.61</td>
<td>2.44%</td>
<td>0.35%</td>
<td>1.95%</td>
</tr>
<tr>
<td>C2_10</td>
<td>289</td>
<td>184434.93</td>
<td>617934.93</td>
<td>303</td>
<td>176348.13</td>
<td>630848.13</td>
<td>4.84%</td>
<td>-4.38%</td>
<td>2.09%</td>
</tr>
<tr>
<td>R1_10</td>
<td>919</td>
<td>486566.98</td>
<td>1865066.98</td>
<td>927</td>
<td>528107.74</td>
<td>1918607.74</td>
<td>0.87%</td>
<td>8.54%</td>
<td>2.87%</td>
</tr>
<tr>
<td>R2_10</td>
<td>190</td>
<td>291992.04</td>
<td>606118.04</td>
<td>190</td>
<td>321118.17</td>
<td>606118.17</td>
<td>0.00%</td>
<td>9.97%</td>
<td>5.05%</td>
</tr>
<tr>
<td>RC1_10</td>
<td>900</td>
<td>452043.29</td>
<td>1802043.29</td>
<td>903</td>
<td>499737.16</td>
<td>1854237.16</td>
<td>0.33%</td>
<td>10.55%</td>
<td>2.90%</td>
</tr>
<tr>
<td>RC2_10</td>
<td>182</td>
<td>277334.49</td>
<td>550734.49</td>
<td>195</td>
<td>258241.86</td>
<td>550741.86</td>
<td>7.14%</td>
<td>-7.02%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Total</td>
<td>3421</td>
<td>213138.54</td>
<td>7262884.56</td>
<td>3482</td>
<td>2223715.67</td>
<td>7446715.67</td>
<td>1.78%</td>
<td>4.33%</td>
<td>2.53%</td>
</tr>
</tbody>
</table>

### ACKNOWLEDGMENTS

The work described in this article was done within the Regio 13 program sponsored by the European Regional Development Fund and by Upper Austrian public funds.

### REFERENCES


