A Developmental Solution to (Dynamic) Capacitated Arc Routing Problems using Genetic Programming

Thomas Weise
Nature Inspired Computation and Applications Laboratory
University of Science and Technology of China (USTC)
Hefei 230027, Anhui, China
tweise@ustc.edu.cn

Alexandre Devert
School of Software Engineering
University of Science and Technology of China (USTC)
Suzhou Industrial Park, Suzhou, Jiangsu, China
marmakoide@yahoo.fr

Ke Tang
Nature Inspired Computation and Applications Laboratory
University of Science and Technology of China (USTC)
Hefei 230027, Anhui, China
ketang@ustc.edu.cn

ABSTRACT

A developmental, ontogenic approach to Capacitated Arc Routing Problems (CARPs) is introduced. The genotypes of this method are constructive heuristics specified as trees of mathematical functions which are evolved with Genetic Programming (GP). In a genotype-phenotype mapping, they guide a virtual vehicle which starts at the depot. The genotype is used to compute a heuristic value for each edge with unsatisfied demands. Local information such as the visiting costs from the current position, the remaining load of the vehicle, and the edge demands are available to the heuristic. The virtual vehicle then serves the edge with the lowest heuristic value and is located at its end. This process is repeated until all requirements have been satisfied. The resulting phenotypes are sets of tours which, in turn, are sequences of edges. We show that our method has three advantages: **1)** The genotypes can be reused to seed the population in new GP runs. **2)** The size of the genotypes is independent from the problem scale. **3)** The evolved heuristics even work well in modified or dynamic scenarios and are robust in the presence of noise.

Categories and Subject Descriptors

G.1.6 [**NUMERICAL ANALYSIS**]: Optimization—*Global Optimization*; G.2.1 [**DISCRETE MATHEMATICS**]: Combinatorics—*Permutations and combinations*; G.3 [**PROBABILITY AND STATISTICS**]: Stochastic processes

Keywords

Capacitated Arc Routing Problem, CARP, Computer-Aided Logistic Planning, Development, Ontogenesis, Genetic Programming, GP, Dynamic Changes, Robustness, Noise

1. INTRODUCTION

Arc Routing Problems (ARPs) are logistic planning problems where the goal is to find optimal routes for vehicles that need to visit certain streets which require some treatment [10]. Classical examples for ARPs are road gritting and salting [18, 19] and the Chinese Postman Problem [13, 17]. In Capacitated Arc Routing Problems (CARPs), the vehicles are initially located at a single depot and have to deliver some product (such as salt or mail) to the roads. This product is available at the depot at a sufficient amount. The vehicles are limited in their capacity, i.e., the amount of product that they can transport. Traversing any road takes a certain time (cost) and the goal is to find tours of minimum cost that satisfy all requirements [27, 28, 33, 36].

CARPs are usually tackled with an optimization algorithm that produces a solution for one specific, static problem instance. Currently, the algorithms obtaining the best results are Memetic Algorithms (MAs) working on a permutation-based search space [28, 33] that *directly* represents the sequence of edges to be visited by a vehicle. They generate new candidate solutions by, e.g., swapping two edges in such a sequential schedule and use *global information* about the impact of these operations on the solution utility.

In this paper, we present a new method for solving CARPs. We deviate from the traditional approaches in two ways. **1)** We utilize an *indirect* representation. The genotypes in our approach are heuristic functions, encoded as trees of mathematical functions in the style well-known from Genetic Programming (GP) [23]. In a *developmental* genotype-phenotype mapping (GPM), the genotypes drive a greedy construction process. This process starts with an empty schedule and iteratively adds the edge which is rated with the lowest heuristic value until all tasks are satisfied.

2) The genotypes, i.e., the evolved heuristic functions, represent the perspective of a truck driver or the Chinese postman – they access *local* information such as the demand of the prospective edges and the costs of serving them from the current location of the vehicle on the road. Decisions therefore do not take into account the global structure of a solution. During the GPM, the overall objective value is not yet known while the edges are added to the solution in a forward-only fashion. The developmental process still results in a phenotype that is a permutation of edges, i.e., a sequential schedule of tasks to perform.

The new approach has three advantages. **1)** Existing genotypes from previous runs can be reused to seed new runs, as they are heuristics to generate solutions. This can lead to faster optimization with better overall results.

2) The size of genotypes is independent of the size of the phenotypes. It is known that the quality of the results obtained by optimization algorithms tends to decrease with a rising number of decision variables. In the context of CARPs, the performance of the solutions obtained through direct encodings will deteriorate with an increasing number n_R of edges to serve. In an indirect representation like ours, however, the size of the genotypes (the function trees) is decoupled from the size of the phenotypes (the schedules) and therefore, independent of the problem scale.

3) The genotypes – although evolved for a specific scenario – can also be applied without further optimization runs to slightly modified scenarios or even different situations. Combined with the truck driver perspective, our method hence is especially suitable for *dynamic* scenarios. In road salting, e.g., one or two roads on a map may become blocked but the overall situation does not change too much. When facing alteration of the scenario used for the optimization, an evolved specialized heuristic therefore will exhibit a graceful performance degradation.

In the next section, we will first give a formal definition of static CARPs. In Section 3, we discuss related work on CARPs, developmental genotype-phenotype mappings, and hyper-heuristics. Our own ontogenic method, based on Genetic Programming, is introduced in Section 4. We then provide the first experimental results achieved with this new method for both static and dynamic CARPs in Section 5. Section 6 concludes the paper with a summary on the current state of our work and our plans for future work.

2. FORMAL PROBLEM DEFINITION

2.1 Problem Instance

Each CARP instance can be fully described by

1. a graph $G = (V, E)$ describing the road network, defined as a tuple of
   (a) a set of vertices V and
   (b) a list $E \subseteq V \times V$ of (directed) edges $e = \overrightarrow{(v_i, v_j)}$ that can be traversed only in one direction, from their starting node *head*$(e) = v_i$ to their ending node *tail*$(e) = v_j$.
2. a cost function $\mathbf{C} : E \mapsto \mathbb{R}^+$ denoting the costs of traversing a road (edge) with a vehicle,
3. a demand function $\mathbf{R} : E \mapsto \mathbb{N}_0$ which assigns the required amount of product to each edge, and
4. the capacity limit $L \in \mathbb{N}_1$ of the vehicle(s) with $e \in E \Rightarrow L \geq \mathbf{R}(e)$.

In many scenarios, there are also undirected edges with requirements. For instance, the direction of the traversal is unimportant when salting a small road. This can be modeled by representing an undirected edge $u = \overline{(v_i, v_j)}$ as two directed edges $\overrightarrow{(v_i, v_j)}$ and $\overrightarrow{(v_j, v_i)}$ in opposite directions. A scenario with n_d directed edges and n_u undirected edges will thus result in an edge list of length $n_e = n_d + 2n_u$, where the first n_d, i.e., e_1 to e_{n_d}, correspond to the existing directed edges. These are followed by the n_u pairs of new directed edges representing the undirected ones, i.e., u_i corresponds to e_{n_d+2i-1} and e_{n_d+2i}, each having the same costs and demands associated as the original undirected edge.

2.2 Solution Space

In CARPs, all edge demands are atomic and cannot be split. This means that the demand of each of the n_d original directed edges must be satisfied during exactly one traversal. Also, for each of the n_u original undirected edges, the demand of exactly one of the two corresponding directed edges must be satisfied in one single step.

As the capacity L of the vehicle is limited and the total demand sum maybe larger than L, this can result in multiple tours, each starting and ending at the depot note $v_1 \in V$.[1] Each tour T corresponds to a list of len(T) edges whose demands should be satisfied.

If two subsequent edges $T[i]$ and $T[i+1]$ of a tour are not connected, i.e., *tail*$(T[i]) \neq$ *head*$(T[i+1])$, the vehicle is assumed to take least-cost route *route*$(T[i]$,$T[i+1])$ between *tail*$(T[i])$ and *head*$(T[i+1])$. The same holds if the starting node of the first edge or the end node of the last edge in a tour, respectively, are not the depot node v_1. In this case, routing costs from/to the depot are added.

A candidate solution x is a set of such tours T and the solution space $\mathbb{X}$ is the set of all possible such candidate solutions. A tour is a permutation of a subset of E. The set of all subsets of E is the power set $\mathcal{P}(E)$. τ be an element of this set and $\mathbf{\Pi}(\tau)$ the set of all possible permutations of the elements of (the edges in) τ. The set $\mathbb{T}$ of all tours is then defined as $\mathbb{T} = \{T : T \in \mathbf{\Pi}(\tau) \wedge \tau \in \mathcal{P}(E)\}$ and the problem space, the set of all possible sets of tours, is given as $\mathbb{X} = \mathcal{P}(\mathbb{T})$.

2.3 Constraints and Objective

The goal of solving a Capacitated Arc Routing Problem is to find a feasible schedule that satisfies all edge demands at the lowest possible costs. For each edge whose demand is to be satisfied, two types of costs arise: the traversal costs along the edge and the routing costs from the current position of the vehicle to the start of the edge. After completing each tour, the vehicle returns back to the depot v_1.

For simplicity's sake, we assume that the list of edges *route*$(e_i$,$e_j)$ describing least-cost path between the end node of any edge e_i and the starting node of any edge e_j is known, as it can easily be calculated with Floyd's Algorithm [15]. The total cost of any edge list r with a vehicle starting at the end of the first edge then is defined as *pathCost* in Equation 1. Additionally, we introduce a special edge, the depot loop $\underline{e} = \overrightarrow{(v_1, v_1)}$, which has zero cost and demand. The total costs *tourCost*(T) of a tour T can be computed by interpreting the tour as an edge list with the depot loop inserted at the beginning and end, as shown in Equation 2.

[1]Alternatively, multiple vehicles may be utilized at once.

Finally, the overall costs of a candidate solution $x \in \mathbb{X}$ and hence, the objective function f, are given as the sum of all tour costs in Equation 3.

$$pathCost(r) = \sum_{i=2}^{\text{len}(r)} \left[\mathbf{C}(r[i]) + \sum_{e \in route(r[i-1], r[i])} \mathbf{C}(e) \right] \quad (1)$$

$$tourCost(T) = pathCost(\underline{e} \circ T \circ \underline{e}) \quad (2)$$

$$f(x) = \sum_{\forall T \in x} tourCost(T) \quad (3)$$

There are three constraints which must be `true` for any candidate solution to be *feasible*. First, the capacity L of the vehicle must not be exceeded at any tour (Equation 4). Each edge may occur at most once in a tour. The constraints c_d and c_u in Equation 5 and Equation 6 specify that any of the original directed edges must occur in exactly one tour and that exactly one of two directed edges representing any of the n_u original undirected edges must occur once as well.

$$c_c(x) = \forall T \in x \Rightarrow \left(\sum_{\forall e \in T} \mathbf{R}(e) \right) \leq L \quad (4)$$

$$c_d(x) = \forall i \in 1..n_d \Rightarrow |\{T : T \in x \land e_i \in T\}| = 1 \quad (5)$$

$$c_u(x) = \forall i \in 1..n_u \Rightarrow |\{T : T \in x \land e_{n_d+2i-1} \in T\}| + |\{T : T \in x \land e_{n_d+2i} \in T\}| = 1 \quad (6)$$

We do not consider limits on the number of tours (or vehicles), as we are interested in low cost solutions only. In cases were such limits are important, one could, e.g., apply a global repair operator similar to [27] discussed in the following section to the phenotypes.

3. RELATED WORK

There are three sets of works that are related to our approach. Here, we will first discuss existing research on CARP followed by research regarding ontogenic representations, and finally concern works on hyper-heuristics.

3.1 CARP

CARPs belong to the class of $\mathcal{NP}$-hard problems [16, 33] which means that algorithms for exact solutions are only feasible for problem instances of small scale or with otherwise special properties. This gives rise to a wide set of heuristic and metaheuristic methods in the area. The Tabu Search approach CARPET [20] represents a candidate solution as set of routes, each of which being a list of vertices annotated with a Boolean variable indicating whether the edge between two vertices should be served or not.

A first Memetic Algorithm (MA) for CARPs, based on a hybridization of a Genetic Algorithm with a local search, is introduced in [24]. Here, the solutions are represented as the sequences in which edges are served and in which intermediate edges, needed for routing purposes only, are omitted. Different from our definition, the solution is only one sequence of tasks which is later divided into separate tours using a heuristic [34].

A deterministic Tabu Search method which can outperform both the CARPET and this MA is introduced in [3]. This algorithm is further improved by introducing a global repair operator which can amend low-cost infeasible solutions [27]. This RTS algorithm is still amongst the best approaches for CARPs.

An Evolutionary Algorithm (EA) for solving CARPs that uses the same representation as defined in Section 2.2 is proposed in [18, 19].

Local search in CARP is usually based on simple operators making small changes to a candidate solution. However, if the search space is large or contains many local optima, bigger changes which are able to leave the basins of attraction of an optimum are necessary. The MA for CARPs (MAENS) given in [33] addressed this problem by introducing a Merge-Split operator which possesses this ability. The experiments in [33] showed that it can outperform all the aforementioned Tabu Search and MA approaches. MAENS has further been extended for solving multi-objective CARPs in [28], where the goal is not only to minimize the total costs, but also the makespan, i.e., the maximum cost over all tours in a candidate solution.

All the above algorithms have in common that they work on a *direct representation* of the solutions. Genotypes and phenotypes are essentially the same and the tours making up a schedule are modified by the search operations directly. The global information that the algorithms utilize comes from the objective functions directly. These functions are not treated as black boxes: knowledge about their structure is, e.g., used in search operations such as 2-opt [24].

Different from that, the Ant Colony Optimization algorithm (ACO) described in [25] combines local and global information in the search. In ACO, (simulated) pheromones correspond to local information that simulated ants use to choose the next step on their path. Based on the overall solution fitness (global information) these pheromones are updated. In [25], initial solutions are furthermore created with heuristics and solutions are refined with local search. This procedure does not allow for reuse of solutions or adaptation to dynamic changes and is more complex than our approach.

3.2 Ontogenic Representation

In *indirect representations*, the search space $\mathbb{G}$ is significantly different from the solution space $\mathbb{X}$ and a genotype-phenotype mapping gpm : $\mathbb{G} \mapsto \mathbb{X}$ translates between them. The dimension of $\mathbb{G}$ may be much smaller than the dimension of $\mathbb{X}$ or even independent from it. Especially the latter case is interesting, as such representations are suitable to solve large-scale problems.

At least two classes of indirect representations may be distinguished [2, 9]: generative and ontogenic approaches. In the generative method, the GPM is a *one-shot* functional mapping from the genotypes to the phenotypes. The GPM may be an arbitrarily complex decoder, but it only uses the information given in the genotypes as input. One example for such mappings in the area of GP is Grammatical Evolution [29, 32], where the genotypes are integer strings and the candidate solutions are sentences of a language defined by a given grammar. Here, the GPM starts with the starting symbol of that static grammar as current variable. The first gene in the genotype identifies the first rule of the grammar fitting to the current variable to be expanded. This may result in new variables occurring, which are then subsequently expanded by rules identified by the following genes.

Ontogenic (or *developmental*) mappings additionally involve feedback from simulations or the process of computing the objective values when building the phenotypes in an *iterative manner* [7, 9]. Our work is a developmental, ontogenic approach which iteratively adds edges to a solution based on an environment's state.

A good example for such a constellation is the experiment described in [14]: The evolution of shapes serving the purpose of heat shields. One shape is made of cells in a grid, one cell is either filled or empty. The decision regarding a cells filling is made by a cellular automaton whose rules are based on the temperature of the current cell and the state of neighboring cells. As the cellular automaton modifies the cells, a simulation of heat diffusion updates the temperature of each cell. The candidate solution is constructed through an *iterative* feedback loop between a construction process and the simulation used to compute a phenotype's fitness.

Recently, it was shown that ontogenic mappings can yield results similar to direct and generative ones but with *lesser* computational effort *despite* the more complex solution creation and evaluation process on the example of the evolution of rigid truss design optimization [9]. In a direct method, the volume of the (up to 600) beams of a truss would be optimized by a numerical optimization algorithm directly. The genotype of a generative approach could be a function which translates the coordinates of a beam to its thickness. In the ontogenic method in [9], the genotypes are functions that receive as parameter the mechanical stress on a beam and return how much the cross section of the beam should be increased. Beginning with a basic beam structure, the mechanical stress is evaluated and the function is applied to each of the beams. The updated truss is simulated again and the process is *iterated* a couple of times. The resulting structure, the phenotype, has up to 600 parameters whereas the genotypes in [9] are multi-layer perceptrons representing the modification function, encoded as real vectors containing, e.g., only 12 neural weights.

3.3 Hyper-Heuristics

Hyper-heuristics [4, 31] are methods with a search space of heuristics. This is also the search space of our method. However, there is a *semantic* difference: hyper-heuristics build heuristics in order to solve a general class of problems whereas our approach searches the perfect heuristic for a fixed scenario. Therefore, we only use one single test case, which also is the training case, at a time.

Hyper-heuristics need to use multiple training cases [5] in order to avoid overfitting to a specific case and test their heuristics on test cases not used during the evolution [21]. Overfitting is acceptable in our approach as it may lead to good approximations of the global optimum. Yet, we will show in the experiments in Section 5 that the heuristics discovered with our GP methods still preserve generality and are efficient for slightly modified or even entirely different problem instances.

4. DEVELOPMENTAL CARP SOLVING

In this paper, we present an ontogenic approach towards CARPs based on Symbolic Regression with GP. The genotypes of our approach are mathematical functions $g : E \mapsto \mathbb{R}$. The functions are expressed as trees of expressions that assign heuristic values to edges and can be considered as constructive heuristics. Each such function is mapped to a candidate solution $x \in \mathbb{X}$ as follows:

1. Start with an empty schedule and
2. locate a simulated vehicle with load L at the depot.
3. The set of unsatisfied edges S is initialized as $S \longleftarrow E$.
4. $\texttt{last}(e) \longleftarrow 0 \; \forall e \in S$.
5. The heuristic function represented by the genotype g is evaluated for each edge $e \in S$.
6. The first edge $e^\star = \arg\min \{g(e) : e \in S\}$ encountered with the lowest heuristic value is chosen to be satisfied next.
7. $\texttt{last}(e) \longleftarrow g(e) \; \forall e \in S$.
8. If the demand $\mathbf{R}(e^\star)$ is larger than the current load in the vehicle *or* $e^\star = \underline{e}$,
   (a) end the current tour,
   (b) return the vehicle to the depot, and set its load back to L.
   (c) Then start a new tour.
9. If $e^\star \neq \underline{e}$, append $e^\star$ to the current tour.
10. Subtract $\mathbf{R}(e^\star)$ from the load of the vehicle.
11. Place the vehicle at point *tail*$(e^\star)$.
12. Add the depot loop to S, i.e., set $S \longleftarrow S \cup \{\underline{e}\}$.
13. Remove $e^\star$ from S (set $S \longleftarrow S \setminus \{e^\star\}$). If $e^\star$ is one of the two directed edges representing an undirected edge u, remove the other directed edge belonging to u as well.
14. If $|S \setminus \{\underline{e}\}| > 0$, go back to Point 5.

The result of this process is a full schedule which fulfills all the constraints introduced in Section 2.3. The genotype-phenotype mapping is ontogenic as it prescribes a developmental process which has access to the following information obtain from simulating the vehicle's behavior:

1. `demand(e)`. The demand $\mathbf{R}(e)$ of the edge divided by the vehicle's capacity, i.e., $\mathbf{R}(e)/L$. For the scaled demand of the depot loop $\underline{e}$, we tried both 0 and -1, which does not lead to significantly different results in the preliminary experiments. We finally chose -1, i.e., the depot loop has an unscaled demand of $-L$.

2. `load`. The remaining amount of the product inside the vehicle divided by the vehicle's capacity L.

3. `cost(e)`. The cost of servicing edge $e \in E$, normalized into $[0, 1]$. These costs contain at least the traversal cost $\mathbf{C}(e)$. If $\texttt{load} \geq \texttt{demand}(e)$, then routing costs from the current location of the vehicle to the starting node *head*(e) of e are added. Otherwise, routing costs are calculated from the current location of the vehicle to the depot v_1 and from there to *head*(e). Normalized into $[0, 1]$ (together with $\texttt{depotCost}(e)$).

4. `depotCost(e)`. The costs to reach the depot from the *end* of e, normalized into $[0, 1]$. Normalization is done by dividing the value by the maximum costs (in terms of both, `cost` and `depotCost`) over all unsatisfied edges.

5. `satisfied`. The fraction of edges with non-zero demands that have already been satisfied $\frac{|E|-|S|}{|E|-1}$.

6. `last(e)`. The heuristic value assigned to the edge e in the previous round of edge selection.

5. EXPERIMENTS

5.1 Benchmark Datasets

We tested our algorithm on five sets of CARP benchmark instances, namely the *gdb* set [6], the *val* set [1], the *egl* set [11, 12, 26], the *br-egl* set [3], and the *kshs* set [22]. These benchmark data sets differ widely in terms of scale, i.e., in the number of vertices, the original number of edges $n_d + n_u$, and the number of tasks $n_R = |\{e : e \in E \land \mathbf{R}(e) > 0\}|$, as well as the number $|x|$ of tours that the lowest-cost solution needs to fulfill all tasks. In Tables 1 to 5, we list the features of these sets as well as the *best* results obtained with these two algorithms from literature (if available) and the corresponding best result achieved with our ontogenic method.

5.2 Experimental Settings

5.2.1 The Two Configurations

Our experiments were conducted with tree-based standard GP using a $(\mu + \lambda)$ population treatment, i.e., truncation selection where μ parents and their λ offspring compete for μ spots in the mating pool, with $\mu = \lambda = 48$. Each run was granted a total of 16 384 function evaluations (FEs). If no improvement in fitness was achieved during at least 1536 FEs (32 generations), an independent restart was performed within the run. The initial population is generated using ramped-half-and-half [23]. Sub-tree exchange crossover, sub-tree replacement mutation, and mathematical simplification (e.g., $a + 0 \longrightarrow a$) were used as search operations in the proportion 2:5:2. The maximum tree depth is set to 10.

The six terminal symbols given in Section 4 were extended with ephemeral random constants [23] and the following purely mathematical functions $a{+}b$, $a{-}b$, $a{*}b$, a/b, $\max\{a, b\}$, $\exp(a)$, $\sin(a)$, $\text{angle}(a, b)$ (where $\text{angle}(a, b)$ returns the angle between the x-coordinate b, the y-coordinate a, and the origin of the Cartesian coordinate system). All functions are protected, i.e., they return -1 instead of $-\infty$, 0 instead of NaN, and 1 instead of $+\infty$. angle, sin, and exp are included in order to provide non-linear transformations.

The above setup will be referred to as GP in the following text. We repeated the experiments done with GP with a second setup (GP*) which is identical with GP except for two modifications: **1)** Instead of using ramped-half-and-half, we seeded the initial population with random individuals chosen from the set of *best* discovered solutions by GP for each of the 81 cases from *gdb* (size 23), *val* (size 34), and *egl* (size 24). **2)** Instead of granting 16 384 FEs, we only granted 8192 (i.e., half as much runtime).

5.2.2 The Objective Function

For simplification purposes, we here define the objective function v used by both of our GP methods directly for a genotype g in Equation 7. Its main component is the total cost f of the plan $x = \text{gpm}(g)$ obtained from the ontogenic GPM applied to g as defined in Equation 3. Pressure towards smaller genotypes is included by subtracting the inverse of $\texttt{weight}(g)$, the number of nodes in the tree g. This penalty only kicks in for two genotypes producing plans of the same costs as it is smaller than 1 and 1 is the unit of costs.

$$v(g) = f(\text{gpm}(g)) - \frac{1}{\texttt{weight}(g)} \tag{7}$$

Table 1: Features of the *gdb* set [6] and obtained results.

Scen. Features			Best Costs (minimize)			
id	n_R	$\lvert x\rvert$	RTS	MAENS	GP	GP*
1	22	5	316	316	*316*	*316*
2	26	6	339	339	*339*	*339*
3	22	5	275	275	*275*	*275*
4	19	4	287	287	*287*	*287*
5	26	6	377	377	383	383
6	22	5	298	298	302	302
7	22	5	325	325	*325*	*325*
8	46	11	348	348	366	363
9	51	11	303	303	320	320
10	25	4	275	275	*275*	*275*
11	45	5	395	395	419	419
12	23	7	458	458	474	474
13	28	7	536	536	558	558
14	21	5	100	100	*100*	*100*
15	21	4	58	58	*58*	*58*
16	28	5	127	127	*127*	*127*
17	28	5	91	91	*91*	*91*
18	36	5	164	164	166	*164*
19	11	3	55	55	*55*	*55*
20	22	5	121	121	123	123
21	33	6	156	156	160	160
22	44	9	200	200	206	206
23	55	11	233	233	239	239

5.3 Experiment 1: Static Optimization

In Tables 1 to 5, we provide the best results achieved during the 30 runs by our two methods in comparison with the (best) results from literature (where also 30 runs are used). It can be seen that the results of both GP and GP* are either the same (*italic* values in the tables) or slightly worse than these currently known lower bounds.

There are three obvious reasons for this: **1)** The results in literature have been obtained by evaluating more candidate solutions (e.g., the MAENS was granted approximately 90 000 FEs [28]). It should be added that our approach has a higher per-solution effort due to the computations necessary in the construction process, but this is still a valid point. **2)** The results from literature are the best ones obtained after years of research and probably correspond to the global optima in many of the benchmark cases. **3)** The indirect representation (and the size of the genotypes) is independent of the problem scale and only encodes a subset of the possible solutions [9]. It therefore can only deliver an approximation of

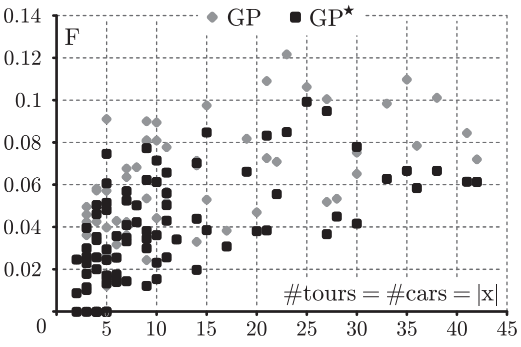


Figure 1: The relative performance F in dependence on the number of tours $|x|$.

Table 2: Features of the *val* set [1] and obtained results.

Scen. Features			Best Costs (minimize)			
id	n_R	$\|x\|$	RTS	MAENS	GP	GP*
1A	39	2	173	173	*173*	*173*
1B	39	4	173	173	181	181
1C	39	9	245	245	251	$\underline{248}$
2A	34	2	227	227	229	229
2B	34	3	259	259	262	262
2C	34	8	457	457	480	480
3A	35	2	81	81	83	83
3B	35	3	87	87	91	$\underline{89}$
3C	35	7	138	138	143	$\underline{140}$
4A	69	3	400	400	417	$\underline{412}$
4B	69	4	412	412	436	$\underline{426}$
4C	69	5	428	428	467	$\underline{460}$
4D	69	9	530	530	573	$\underline{571}$
5A	65	3	423	423	444	$\underline{434}$
5B	65	4	446	446	465	$\underline{455}$
5C	65	5	474	474	497	$\underline{488}$
5D	65	9	583	577	629	$\underline{613}$
6A	50	3	223	223	229	$\underline{227}$
6B	50	4	233	233	239	239
6C	50	11	317	317	335	$\underline{333}$
7A	66	6	279	279	291	$\underline{289}$
7B	66	6	283	283	292	$\underline{287}$
7C	66	11	334	334	360	$\underline{356}$
8A	63	3	386	386	400	$\underline{390}$
8B	63	4	395	395	415	$\underline{409}$
8C	63	9	524	521	563	$\underline{539}$
9A	92	3	323	323	331	331
9B	92	5	326	326	339	$\underline{334}$
9C	92	5	332	332	351	$\underline{348}$
9D	92	10	391	391	426	$\underline{415}$
10A	97	3	428	428	445	445
10B	97	4	436	436	461	$\underline{458}$
10C	97	5	446	446	469	469
10D	97	10	534	531	574	$\underline{569}$

Table 3: Features of the *egl* set [11, 12, 26] and obtained results.

Scen. Features			Best Costs (minimize)			
id	n_R	$\|x\|$	RTS	MAENS	GP	GP*
E1-A	98	5	3548	3548	3644	$\underline{3609}$
E1-B	98	7	4498	4498	4689	$\underline{4648}$
E1-C	98	10	5595	5595	5798	$\underline{5725}$
E2-A	98	7	5018	5018	5338	$\underline{5282}$
E2-B	98	10	6317	6317	6704	$\underline{6546}$
E2-C	98	15	8335	8335	8777	$\underline{8657}$
E3-A	98	8	5898	5898	6301	$\underline{6148}$
E3-B	98	12	7787	7775	8041	8041
E3-C	98	17	10 305	10 292	10 687	$\underline{10\,610}$
E4-A	98	9	6461	6456	6802	$\underline{6703}$
E4-B	98	14	9026	8998	9621	$\underline{9394}$
E4-C	98	20	11 598	11 561	12 105	$\underline{12\,002}$
S1-A	190	7	5018	5018	5358	$\underline{5304}$
S1-B	190	10	6394	6388	6671	$\underline{6487}$
S1-C	190	14	8518	8518	8800	$\underline{8687}$
S2-A	190	14	9970	9895	10 602	$\underline{10\,591}$
S2-B	190	21	13 345	13 147	14 102	$\underline{13\,653}$
S2-C	190	28	16 600	16 430	17 309	$\underline{17\,170}$
S3-A	190	15	10 284	10 257	11 258	$\underline{11\,127}$
S3-B	190	22	13 857	13 749	14 725	$\underline{14\,514}$
S3-C	190	30	17 316	17 207	18 330	$\underline{17\,925}$
S4-A	190	19	12 348	12 341	13 351	$\underline{13\,159}$
S4-B	190	27	16 442	16 337	17 186	$\underline{16\,937}$
S4-C	190	36	20 281	20 538	21 874	$\underline{21\,467}$

Table 4: Features of the *br-egl* set [3] and obtained results.

Scenario Features			Best Costs (minimize)		
id	n_R	$\|x\|$	RTS	GP	GP*
G1-A	375	21	1 025 765	1 137 696	$\underline{1\,111\,244}$
G1-B	375	25	1 135 873	1 256 632	$\underline{1\,248\,726}$
G1-C	375	30	1 271 894	$\underline{1\,367\,675}$	1 371 167
G1-D	375	35	1 402 433	1 556 484	$\underline{1\,495\,953}$
G1-E	375	41	1 558 548	1 690 310	$\underline{1\,654\,457}$
G2-A	375	23	1 125 602	1 262 630	$\underline{1\,221\,212}$
G2-B	375	27	1 242 542	1 367 509	$\underline{1\,360\,481}$
G2-C	375	33	1 401 583	1 539 577	$\underline{1\,489\,739}$
G2-D	375	38	1 516 072	1 669 510	$\underline{1\,617\,203}$
G2-E	375	42	1 668 348	1 788 617	$\underline{1\,770\,868}$

Table 5: Features of the *kshs* set [22] and obtained results.

Scen. Feat.			Best Costs	
id	n_R	$\|x\|$	GP	GP*
1	30	4	14 661	14 661
2	30	4	9863	9863
3	30	4	9320	9320
4	30	4	11 684	$\underline{11\,498}$
5	30	3	10 957	10 957
6	30	3	10 197	10 197

the global optimum by nature. Direct methods like MAENS, at least in theory, always can discover the global optimum if given enough runtime.

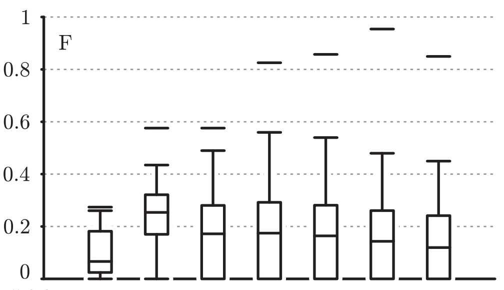
Let us now compare GP and GP*. By seeding the population and providing half of the FEs only, we would expect a convergence to one of the previously discovered good solutions or worse results (as $\mu = 48$ is less than the number of candidate genotypes for seeding). However, what we observe instead is that we obtain better results in many scenarios ($\underline{\text{underlined}}$ elements the tables). This means that the genotypes, i.e., the evolved case-specific constructive heuristics, are general or at least contain general modules that can be combined in order to form even better heuristics.

The most astonishing results here are for sets *br-egl* and *kshs* in Tables 4 and 5. For seeding the GP*, only results for the sets *val*, *gdb*, and *egl* were used. As the sets *br-egl* and *kshs* contain entirely different benchmark cases, we would expect the GP* with its lower FE contingent to be in a stark disadvantage. However, the opposite is the case: In *all but one* of the *br-egl* and one of the *kshs* cases, it performs better than GP. There is only a single case where GP is better. This is especially remarkable as the instances in *br-egl* are of much larger scale (see the n_R and $|x|$ values) than any in the other benchmark sets. The cases in *kshs*, on the other hand, are of a smaller scale than most other instances.

The data of this experiment allowed us to identify which feature of a problem instance has the strongest impact on the solution quality that our method can provide.

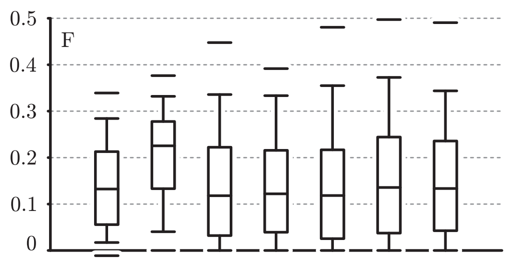
In Figure 1, we plot the relative performance F of the best result x obtained with our approaches in comparison with the global optimum $x^\star$ from literature for each benchmark instance as defined in Equation 8, in relation to the number $|x|$ of tours T in x. The more tours are necessary, the higher does F tend to be.

$$F = \frac{f(x)}{f(x^\star)} - 1 \qquad (8)$$



#del	1	2	3	4	5	6	7
#cases	25	271	593	595	570	302	136
#runs	175	1897	4151	4165	3990	2114	952

Figure 2: Box plot: static heuristic vs. GP* in the *gdb10* scenarios.



#del	1	2	3	4	5	6	7
#cases	66	214	198	192	186	181	175
#runs	462	1498	1386	1344	1302	1267	1225

Figure 3: Box plot: static heuristic vs. GP* in the *val7C* scenarios.

This is natural, as the genotypes are constructive heuristics and, to some degree, greedy methods. As said in the first paragraph of Section 2.3, each tour also contains the final travel back to the depot. An ideal heuristic should therefore select edges closer to the depot near the end of a tour in order to minimize the return costs. For a forward-only heuristic, switching between cheap edges from the current location towards such edges at the right time seems to be complicated – and our method also suffers from this problem. The more tours a solution consists of, the more often this problem occurs and, hence, the higher F, i.e., the farther the solutions are away from the global optimum. Instead of the problem scale n_R, this trend is the most determining factor in our method.

5.4 Experiment 2: Dynamic Optimization & Noise Robustness

Seeding the population in GP* with previous solutions showed that evolved genotypes (heuristics) can be used as basis to find good heuristics for new scenarios. In this second experiment, we want to test how a genotype can deal with limited changes of the scenario for which it was synthesized.

We choose case 10 from set *gdb* (*gdb10*) and case 7C from *val* (*val7C*) as basis for this experiment. *gdb10* has a smaller scale of $n_R = 25$ and RTS, MAENS, as well as our GP and GP* all repetitively yielded solutions with costs 275 and needed 4 tours. *val*7C, on the other hand, has a larger scale of $n_R = 66$. Here, our methods found a worse result (costs 356, 11 tours) than RTS and MAENS (costs 334). This solution has only been discovered once (by GP*) and is hence not in the seeds for GP*. The two basic scenarios are therefore quite different.

From these scenarios, we derive new scenarios, each with a certain number *#del* of edges removed. We then compare the solutions provided by a fixed heuristic with those provided by GP*. For each value of *#del* in $1 \dots 7$, we created a number *#cases* of solvable scenarios (with connected graphs). As fixed heuristic, we choose the first genotype g with costs 275 discovered by GP* in the previous experiment for *gdb10* and the one solution with costs 356 for *val7C*. We performed 7 runs with GP* for each of these scenarios.

In Figures 2 and 3, we give box plots of the relative performance F of the static heuristics g compared with the *best* results delivered by the 7 GP* runs (analogously to Equation 8). For each of the *#cases* scenarios per *#del* setting, one F value is obtained. All the *#cases* values for one *#del* setting together make up one box with whiskers at the 0.025 and 0.975 quantile and markers for the maximum and minimum.

The graphs for *gdb10* and *val7C* exhibit similar performance changes. However, the worst case scenarios (top whiskers) in the larger graph *val7C* are generally better: deleting a few edges has less impact than for the smaller graph *gdb10*. The most interesting observation is that the relative performance of the static heuristic in median (middle line of the boxes) is only for the *#del* $= 2$ case more than 20% worse than the GP* result – for both, *gdb10* and *val7C*. In all other situations, we observe a gentle performance degeneration and even a stabilization for rising *#del*. In fact, in five out of the seven *#del* settings, the 25% quantile is 0 for *gdb10* – and $F = 0$ means that g and GP* have identical performance.

20% performance loss towards a near-optimal solution may be acceptable if pervious heavily relied-on connections are removed from the graph and quick decisions need to be made. Yet, we here would prefer a "readjustment" run with GP* and a seeded population instead of only re-using the genotypes as is.

Besides the dynamic aspect, this experiment can also be considered as proof for the noise robustness of our method. Such behavior has been found in many works on ontogenic mappings such as [14] and likely explanations for it are given in [8].

6. CONCLUSIONS AND FUTURE WORK

With the results in this paper, we have demonstrated that a developmental approach to the Capacitated Arc Routing Problem is feasible for both static and dynamic cases. We have shown that Genetic Programming is a suitable method for evolving a heuristic function for this purpose.

Compared to direct encodings, our experiments demonstrated three advantages: **1)** The size of the genotypes is independent from the problem scale. The penalty in terms of the objective value that we pay for this and for using only local information is small. **2)** Good genotypes from prior runs can be reused to seed the population of GP even for entirely different scenarios. **3)** The genotypes represent heuristics which can even deliver good results in case of changes to a scenario. They may also be the basis for cheap "readjustment" optimization runs similar to our GP* method.

The most promising results were obtained with method GP*, Genetic Programming with a seeded population. This showed that even specialized heuristics from different scenarios are likely composed of general building blocks which can be combined and refined in a short optimization run.

Our plans for future work concern two main points. **1)** In [9], the function driving the development in the ontogenic synthesis of beam data structures is represented as a multi-layer perceptron (MLP) whose weight vector is evolved with a CMA-ES variant [30]. This is also possible in our scenario and we will run corresponding experiments in the near future in order to find out whether GP or MLPs are more efficient in this scenario. We will also test whether such a "smooth" representation may offer better reuse and adaptation capabilities or not.

2) The approach to CARPs introduced in this paper is applicable to other Vehicle Routing Problems as well. Therefore, we will conduct experiments on different classes of computer-aided logistic planning.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of Anhui Province under Grant 1108085J16, the Chinese Academy of Sciences Fellowship for Young International Scientists CX05040000001, the National Natural Science Foundation of China under Grant 61150110488, and by a Special Financial Grant from the China Postdoctoral Science Foundation (number 201104329).

References

1. E. Benavent, V. Campos, A. Corberán, and E. Mota. The Capacitated Arc Routing Problem. Lower Bounds. *Networks*, 22(7):669–690, December 1992. doi: 10.1002/net.3230220706.
2. P. J. Bentley and S. P. Kumar. The Ways to Grow Designs: A Comparison of Embryogenies for an Evolutionary Design Problem. In W. Banzhaf, J. M. Daida, Á. E. Eiben, M. H. Garzon, V. Honavar, M. J. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99)*, pages 35–43. Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1999. URL `http://www.cs.ucl.ac.uk/staff/ucacpjb/BEKUC1.pdf`.
3. J. Brandão and R. W. Eglese. A Deterministic Tabu Search Algorithm for the Capacitated Arc Routing Problem. *Computers & Operations Research*, 35 (4):1112–1126, April 2008. doi: 10.1016/j.cor.2006.07.007. URL `http://eprints.lancs.ac.uk/48773/1/Document.pdf`. Also: Lancaster University Management School Working Paper 2005/027.
4. E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu. A Survey of Hyper-Heuristics. Computer Science Technical Report NOTTCS-TR-SUB-0906241418-2747, University of Nottingham, School of Computer Science & Information Technology: Nottingham, UK, March 2009. URL `http://www.cs.nott.ac.uk/TR/SUB/SUB-0906241418-2747.pdf`.
5. E. K. Burke, M. Hyde, G. Kendall, and J. R. Woodward. A Genetic Programming Hyper-Heuristic Approach for Evolving Two Dimensional Strip Packing Heuristics. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 14(6):942–958, December 2010. doi: 10.1109/TEVC.2010.2041061. URL `http://www.genetic-programming.org/hc2011/02-Burke/Burke-Paper.pdf`.
6. J. S. DeArmon. A Comparison of Heuristics for the Capacitated Chinese Postman Problem. Master's thesis, University of Maryland: College Park, MD, USA, 1981.
7. A. Devert. *Building Processes Optimization: Toward an Artificial Ontogeny based Approach.* PhD thesis, Université Paris-Sud, Ecole Doctorale d'Informatique: Paris, France and Institut National de Recherche en Informatique et en Automatique (INRIA), Centre de Recherche Saclay – Île-de-France: Orsay, France, May 2009.
8. A. Devert, N. Bredèche, and M. Schoenauer. Robustness and the Halting Problem for Multicellular Artificial Ontogeny. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 15(3):387–404, June 2011. doi: 10.1109/TEVC.2011.2125969. URL `http://www.marmakoide.org/download/publications/devbresch-ieeetevocomp-2011.pdf`.
9. A. Devert, T. Weise, and K. Táng. A Study on Scalable Representations for Evolutionary Optimization of Ground Structures. *Evolutionary Computation*, 20, 2012. doi: 10.1162/EVCO_a_00054. URL `http://www.marmakoide.org/download/publications/devweita-ecj-preprint.pdf`.
10. M. Dror, editor. *Arc Routing: Theory, Solutions and Applications.* Springer-Verlag GmbH: Berlin, Germany, 2000. ISBN `0-7923-7898-9`.
11. R. W. Eglese. Routing Winter Gritting Vehicles. *Discrete Applied Mathematics – The Journal of Combinatorial Algorithms, Informatics and Computational Sciences*, 48(3):231–244, February 15, 1994. doi: 10.1016/0166-218X(92)00003-5.
12. R. W. Eglese and L. Y. O. Li. A Tabu Search based Heuristic for Arc Routing with a Capacity Constraint and Time Deadline. In I. H. Osman and J. P. Kelly, editors, *Meta-heuristics Theory and Applications*, pages 633–649. Kluwer Academic Publishers: Norwell, MA, USA, 1996.
13. H. A. Eiselt, M. Gendrau, and G. Laporte. Arc Routing Problems, Part I: The Chinese Postman Problem. *Operations Research*, 43(2):231–242, March–April 1995.
14. N. S. Estévez and H. Lipson. Dynamical Blueprints: Exploiting Levels of System-Environment Interaction. In D. Thierens, H. Beyer, J. C. Bongard, J. Branke, J. A. Clark, D. Cliff, C. B. Congdon, K. Deb, B. Doerr, T. Kovacs, S. P. Kumar, J. F. Miller, J. H. Moore, F. Neumann, M. Pelikan, R. Poli, K. Sastry, K. O. Stanley, T. Stützle, R. A. Watson, and I. Wegener, editors, *Proceedings of 9th Genetic and Evolutionary Computation Conference (GECCO'07)*, pages 238–244. ACM Press: New York, NY, USA, 2007. doi: 10.1145/1276958.1277009. URL `http://ccsl.mae.cornell.edu/papers/gecco07_estevez.pdf`.
15. R. W. Floyd. Algorithm 97 (SHORTEST PATH). *Communications of the ACM (CACM)*, 5(6):345, June 1, 1962. doi: 10.1145/367766.368168.
16. B. L. Golden and R. T. Wong. Capacitated Arc Routing Problems. *Networks*, 11(3):305–315, 1981. doi: 10.1002/net.3230110308.
17. M. Guăn. Graphic Programming Using Odd or Even Points. *Chinese Mathematics*, 1:273–277, 1962.
18. H. Handa, L. Chapman, and X. Yáo. Robust Route Optimization for Gritting/Salting Trucks: A CERCIA Experience. *IEEE Computational Intelligence Magazine*, 1(1):6–9, February 2006. doi: 10.1109/MCI.

2006.1597056. URL http://www.cs.bham.ac.uk/~xin/papers/SaltingCIMagazine.pdf.

19. H. Handa, D. Lin, L. Chapman, and X. Yáo. Robust Solution of Salting Route Optimisation Using Evolutionary Algorithms. In G. G. Yen, S. M. Lucas, G. B. Fogel, G. Kendall, R. Salomon, B. Zhang, C. A. Coello Coello, and T. P. Runarsson, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'06), 2006 IEEE World Congress on Computation Intelligence (WCCI'06)*, pages 3098–3105. IEEE Computer Society: Piscataway, NJ, USA, IEEE Computer Society: Piscataway, NJ, USA, 2006. doi: 10.1109/CEC.2006.1688701. URL http://escholarship.lib.okayama-u.ac.jp/industrial_engineering/2/.
20. A. Hertz, G. Laporte, and M. Mittaz. A Tabu Search Heuristic for the Capacitated Arc Routing Problem. *Operations Research*, 48(1):129–135, January–February 2000. doi: 10.1287/opre.48.1.129.12455.
21. D. Jakobović and L. Budin. Dynamic Scheduling with Genetic Programming. In P. Collet, M. Tomassini, M. Ebner, S. M. Gustafson, and A. Ekárt, editors, *Proceedings of the 9th European Conference Genetic Programming (EuroGP'06)*, volume 3905/2006 of *Theoretical Computer Science and General Issues (SL 1), Lecture Notes in Computer Science (LNCS)*, pages 73–84. Springer-Verlag GmbH: Berlin, Germany, 2006. doi: 10.1007/11729976_7. URL http://gp.zemris.fer.hr/scheduling/EuroGP_2006.pdf.
22. M. Kiuchi, Y. Shinano, R. Hirabayashi, and Y. Saruwatari. An Exact Algorithm for the Capacitated Arc Routing Problem using Parallel Branch and Bound Method. In *Abstracts of the National Conference of the Operations Research Society of Japan*, pages 28–29. Operations Research Society of Japan (OSRJ): Tōkyō, Japan, 1995. Written in Japanese.
23. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Bradford Books. MIT Press: Cambridge, MA, USA, December 1992. ISBN 0-262-11170-5. 1992 first edition, 1993 second edition.
24. P. Lacomme, C. Prins, and W. Ramdane-Chérif. Competitive Memetic Algorithms for Arc Routing Problems. *Annals of Operations Research*, 131(1-4):159–185, October 2004. doi: 10.1023/B:ANOR.0000039517.35989.6d. URL http://seisco-outil-calcul-reparti.googlecode.com/svn/trunk/documentation/Documentation/memeticalgorithm.pdf.
25. P. Lacomme, C. Prins, and A. Tanguy. First Competitive Ant Colony Scheme for the CARP. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS'04)*, volume 3172/2004 of *Lecture Notes in Computer Science (LNCS)*, pages 426–427. Springer-Verlag GmbH: Berlin, Germany, 2004. doi: 10.1007/978-3-540-28646-2_48.
26. L. Y. O. Li and R. W. Eglese. An Interactive Algorithm for Vehicle Routeing for Winter - Gritting. *The Journal of the Operational Research Society (JORS)*, 47(2):217–228, February 1996.
27. Y. Méi, K. Táng, and X. Yáo. A Global Repair Operator for Capacitated Arc Routing Problem. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 39(3):723–734, June 2009. doi: 10.1109/TSMCB.2008.2008906. URL http://www.cs.bham.ac.uk/~xin/papers/TSMC09_MeiTangYao.pdf.
28. Y. Méi, K. Táng, and X. Yáo. Decomposition-Based Memetic Algorithm for Multi-Objective Capacitated Arc Routing Problem. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 15 (2):151–165, April 2011. doi: 10.1109/TEVC.2010.2051446. URL http://ieee-cis.org/_files/EAC_Research_2009_Report_Mei.pdf.
29. M. O'Neill and C. Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*, volume 4 of *Genetic Programming Series*. Springer Science+Business Media, Inc.: New York, NY, USA, 2003. ISBN 1402074441.
30. R. Ros and N. Hansen. A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity. In *Proceedings of 10th International Conference on Parallel Problem Solving from Nature (PPSN X)*, volume 5199/2008 of *Theoretical Computer Science and General Issues (SL 1), Lecture Notes in Computer Science (LNCS)*, pages 296–305. Springer-Verlag GmbH: Berlin, Germany, 2008. doi: 10.1007/978-3-540-87700-4_30. URL http://hal.inria.fr/docs/00/27/32/71/PS/RR-6498.ps.
31. P. Ross. Hyper-Heuristics. In E. K. Burke and G. Kendall, editors, *Search Methodologies – Introductory Tutorials in Optimization and Decision Support Techniques*, chapter 17, pages 529–556. Springer Science+Business Media, Inc.: New York, NY, USA, 2005. doi: 10.1007/0-387-28356-0_17.
32. C. Ryan, J. J. Collins, and M. O'Neill. Grammatical Evolution: Evolving Programs for an Arbitrary Language. In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, editors, *Proceedings of the First European Workshop on Genetic Programming (EuroGP'98)*, volume 1391/1998 of *Lecture Notes in Computer Science (LNCS)*, pages 83–95. Springer-Verlag GmbH: Berlin, Germany, 1998. URL http://www.grammatical-evolution.org/papers/eurogp98.ps.
33. K. Táng, Y. Méi, and X. Yáo. Memetic Algorithm with Extended Neighborhood Search for Capacitated Arc Routing Problems. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 13(5):1151–1166, October 2009. doi: 10.1109/TEVC.2009.2023449. URL http://staff.ustc.edu.cn/~ketang/papers/TangMeiYao_TEVC09.pdf.
34. G. Ulusoy. The Fleet Size and Mix Problem for Capacitated Arc Routing. *European Journal of Operational Research (EJOR)*, 22(3):329–337, December 1985. doi: 10.1016/0377-2217(85)90252-8.
35. T. Weise, K. Táng, and A. Devert. A Developmental Solution to (Dynamic) Capacitated Arc Routing Problems using Genetic Programming. In T. Soule and J. H. Moore, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'12)*, pages 831–838. Association for Computing Machinery (ACM): New York, NY, USA, 2012. doi: 10.1145/2330163.2330278.
36. S. Wøhlk. A Decade of Capacitated Arc Routing. In B. L. Golden, S. R. Raghavan, and E. A. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Re-*

search/Computer Science Interfaces Series, chapter Part I / Overviews and Surveys, Chapter 2, pages 29–48. Springer-Verlag GmbH: Berlin, Germany, 2008. doi: 10.1007/978-0-387-77778-8_2. URL http://www.cs.uwaterloo.ca/~msemamit/A%20Decade%20of%20Capacitated%20Arc%20Routing.pdf.

```
@inproceedings{WDT2012ADSTDCARPUGP,
 author   = {Thomas Weise and Ke Tang and Alexandre Devert},
 title    = {A Developmental Solution to (Dynamic)
             Capacitated Arc Routing Problems using
             Genetic Programming},
 booktitle= {Proceedings of the 14th Genetic and
             Evolutionary Computation Conference (GECCO'12)},
 publisher= {ACM Press: {New York, NY, USA}},
 year     = {2012},
 month    = jul # {7--11,~},
 location = {Philadelphia, PA, USA},
 pages    = {831--838},
 doi      = {10.1145/2330163.2330278},
 isbn     = {978-1-4503-1177-9},
},
```