To appear in Foundations of Computing and Decision Sciences, Vol. 29, No. 3, 2004.

SYSTEMATIC CONSTRUCTION OF RECOMBINATION OPERATORS FOR THE VEHICLE ROUTING PROBLEM

Marek KUBIAK*

Abstract. The paper presents a way of systematic construction of recombination operators for the vehicle routing problem by global convexity tests. It describes the vehicle routing problem and defines similarity measures for its solutions, and presents the results of global convexity tests; it has been demonstrated here that strong global convexity exists in almost all instances of the problem. Based on results of the tests some distance preserving recombination operators are defined. Computational experiments with a genetic local search indicate that these operators significantly ameliorate the search process and generate very good solutions. It is also shown that preservation, by an operator, of both of important features of solutions found in global convexity tests speeds up computation. The paper is, therefore, another evidence that systematic construction of recombination operators is a good means of adaptation of the genetic local search to a problem.

1. Introduction

Metaheuristic algorithms has been often and successfully used, at present, as means of solving diverse combinatorial optimisation problems. It must be noted, however, that a metaheuristic algorithm does not have to be efficient in solving all kinds of problems. Moreover, the No Free Lunch theorem ([25]) states that there is no general optimisation method that performs better than any other, including a complete enumeration of solutions of a problem, if performance is averaged over all possible objective functions (i.e. problems and their instances). Therefore one should be well aware of the fact that a certain optimisation method (e.g. a metaheuristic algorithm) performs better than others only under some, probably implicit, assumptions about the nature of the problem being solved. Different metaheuristics may vary with respect to these assumptions, which means that the algorithm

*Poznań University of Technology, Institute of Computing Science, Piotrowo 3A, 60-965 Poznań, Poland. E-mail: Marek.Kubiak@cs.put.poznan.pl.

for a particular problem should be chosen carefully and rationally. The choice should not be influenced by personal bias.

Arguably, these assumptions of algorithms are met by practical problems, since metaheuristics are often successfully used in optimisation, and only artificially formulated problems do not fulfill them. It must be remembered, however, that metaheuristics are only schemes of algorithms that differ from problem to problem or even from implementation to implementation (for the same problem) by their adaptation. Elements of this adaptation, e.g. neighbourhood operators in a local search or recombination operators in an evolutionary algorithm, may strongly influence the degree in which assumptions of a metaheuristic are met. This is why it has been stated in [9]:

> the way a given metaheuristic is adapted to a particular problem may have a crucial influence on its performance.

This statement may be reinforced by an experiment conducted by Michalewicz for a set of certain metaheuristics, namely evolutionary algorithms ([17], [18]). The experiment showed that an algorithm reaches maximum efficiency if it is totally adapted to a problem. Michalewicz concludes with words of Davis ([5]) that if we shrink from adding problem-specific knowledge to genetic algorithms it is very likely that they are worse than any reasonable optimisation algorithm which takes such knowledge into account.

The above considerations may be thus summarised in two short statements:

- the metaheurstic algorithms for a particular problem should be chosen carefully and rationally, based on its (hopefully known) assumptions and properties,
- the chosen algrorithm should be systematically adapted to the problem: problem-specific knowledge must be introduced to increase efficiency.

This paper presents a procedure of systematic construction of recombination operators based on global convexity tests as a method of such adaptation of the genetic local search (GLS) to the Vehicle Routing Problem (VRP). It shows that this construction is a good means of adaptation and results in a very good algorithm.

The paper is organised in the following manner: section 2 describes the metaheuristic algorithm used (GLS). The following one explains why this algorithm may be suitable to solve VRP; it also describes global convexity as a basis of adaptation of GLS to the problem. Section 4 enumerates methods of adaptation of a metaheuristic to a problem, while section 5 presents one of them, that is systematic construction of recombination operators, in detail. In section 6 VRP is formally defined and its used instances mentioned. Next three sections present the whole process of the systematic construction of operators for VRP: definitions of similarity measures (section 7), results of global convexity tests (section 8) and definitions of four recombination operators (section 9). Section 10 describes two computational experiments conducted in order to evaluate GLS with new operators. The last section presents conclusions and outlines further research.

2. Genetic Local Search

The metaheuristic algorithm used in this work to solve the Vehicle Routing Problem (which is defined later in the paper) is the genetic local search, also called the memetic algorithm or the hybrid genetic algorithm ([9], [13]). It is an evolutionary algorithm which also incorporates a local search procedure (LS). One of possible general descriptions of this algorithm (very similar to that presented in [9]) is as follows (K denotes the size of the population):

$GeneticLocalSearch()$

create an empty population
for $i = 1, \ldots, K$ do:
create an initial solution s_i
perform LS on s_i; s'_i is the result
add s'_i to the population
do:
chose randomly (with uniform probability) two different solutions s_1 and s_2 from the population
perform a crossover on s_1 and s_2; s_3 is the offspring
perform LS on s_3; s'_3 is the result
if (s'_3 is better than the worst solution in the population and s'_3 is different from all in the population) then add s'_3 and remove the worst solution
choose randomly (with uniform probability) one solution s_4 from the population
perform a mutation on s_4; s_5 is the result
perform LS on s_5; s'_5 is the result
if (s'_5 is better than the worst solution in the population and s'_5 is different from all in the population) then add s'_5 and remove the worst solution
while a stopping criterion is not met

The main difference from the algorithm used in [9] is one compulsory mutation in each iteration of GLS.

As it has been said in [14]:

> the general idea behind memetic algorithms is to combine the advantages of evolutionary operators that determine interesting regions of the search space with local neighbourhood search that quickly finds good solutions in a small region of the search space.

3. Why GLS?

But however clearly a researcher's intuition about the algorithm is formulated, it does not suffice, given the introductory remarks, to justify the choice of this particular metaheuristic. Why the genetic local search?

There are three main premises that justify the application of this metaheuristic to VRP:

- the feature of a problem that makes GLS work efficiently on it is known,
- some very efficient GLS algorithms have been constructed for different problems and these algorithms have been adapted to the problems based on this very feature,
- VRP is very similar to the travelling salesman problem (TSP), which has been proven to possess this particular feature (at least in some instances).

This feature of a problem (or, to be strict, of an instance of a problem) is called global convexity. It is a phenomenon occurring in the fitness landscape ([2], [10], [9]) of an instance: solutions become better as the distance between them decreases (with respect to some distance measure). Global convexity also assumes that global optima are roughly in the centre of the 'big-valley' structure ([2]). Figure 1 (similar to the one of Boese) helps to visualise global convexity. It shows that although neighbours in the fitness landscape do not have to preserve strict convexity (so there are local optima) there is a tendency of convexity on the global scale.

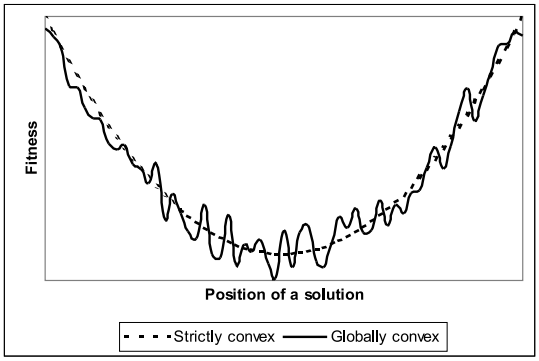

Figure 1. Comparison of strictly and globally convex fitness landscapes.

However, since global convexity is only a tendency, it may be measured, for example, by means of statistics and occur only up to a certain degree. It is also important that this feature of the fitness landscape is very much dependent on the definition of distance (or similarity) used, because this distance determines mutual positions of points (solutions) in the landscape. While for some definitions of distance global convexity may exist, for others there may be not even a trace of it. Another important factor is the instance itself; some instances of a problem may have globally convex landscape and for some others it may be not revealed.

The importance of global convexity for genetic algorithms (GA) was investigated by Jones and Forrest ([10]). They used a notion of 'the relationship between fitness and distance to the goal' (a global optimum), which is equivalent to global convexity. As a measure of this relationship they used the value of correlation coefficient between fitness and distance to a global optimum (the fitness-distance correlation, FDC) for large sets of solutions. They also examined this relationship on scatter plots of fitness and distance, and concluded that FDC values were very useful in predicting a problem difficulty for GA, but sometimes they did not reveal the existence of a 'path to the goal'. So they advised to examine also the mentioned plots 'in case where there is a structure in this relationship that cannot be detected by correlation'.

The results obtained in [10] are applicable to GLS as well, because it is very similar to GA. The memetic algorithm uses exactly the same mechanisms as GA (works on a population with recombination and selection) and the sole difference between the two is that GLS works only on local optima encoded in some problem-specific representation.

Another important fact is that Jones and Forrest used only binary representation of solutions and Hamming distance. Therefore, as they admit, their results may be only an approximation and

> FDC would be presumably more accurate if it were based on the distance between points *according to the operator in use by the algorithm*.

Thus, using a distance measure which takes the actually used recombination operators into consideration would, in their opinion, ameliorate the results even more.

This conclusion was used by Merz and Freisleben ([13], [14], [15], [16]), and Jaszkiewicz and Kominek ([9]) to propose a method of systematic and justified adaptation of GLS to a problem. Based on results of global convexity tests and definition of distance they implemented the so-called distance preserving crossovers (DPX) which appeared extremely efficient in GLS algorithms for TSP, the quadratic assignment problem (QAP), the graph bi-partitioning problem (GBP) and a VRP. These results strongly indicate that systematic procedure of adaptation of the memetic algorithm (i.e. definition of recombination operators based on results of global convexity tests) is a very promising and also a justified method of construction of such an algorithm, if only global convexity is revealed in instances of the problem being solved.

An attempt to use GLS to solve the Vehicle Routing Problem is last but not least justified by the similarity of VRP and TSP. The former is just a slight modification (generalisation) of the latter and in instances of TSP global convexity was revealed ([2]), which resulted in implementation of a very efficient local search. In his experiment, Boese used a distance measure which counted different edges in two solutions; the measure may be easily adapted to VRP solutions.

4. Methods of systematic adaptation of metaheuristics to a problem

Only the procedure of systematic construction of recombination operators based on results of global convexity tests will be used in this discussion. There are, however, other methods of justified adaptation of metaheuristics to a problem. These include:

- construction (or choice) of neighourhood operators for a local search procedure based on random walk correlation length ([13], [22], [24]),
- construction of a mutation operator for an evolutionary algorithm based on the size of an atractor of local optima ([13]),
- forbidding such movements in LS procedure of GLS which change features of solutions preserved by crossover operators ([8]).

All these methods, although may be quite efficient, are not considered here as they are beyond the scope of this paper.

5. Systematic construction of recombination operators

The procedure is as follows:

1. Define some distance (or similarity) measure which, hopefully, influences quality of solutions of the problem being solved.
2. Generate a diversified sample of good solutions of all instances of the problem.
3. Compute FDC in these samples and also analyse fitness-distance plots.
4. If global convexity is revealed implement recombination operators preserving the defined distance.

The success of this procedure is entirely dependent on results of global convexity tests. Only if convexity is revealed distance preserving operators are expected to be efficient.

One part of this procedure requires a more detailed description, namely the computation of FDC. Generally, the correlation coefficient between two features $u(s)$ and $w(s)$ of objects from a random sample $S_n = (s_1, \ldots, s_n)$ is defined as:

$$\hat{\rho}(u, w) = \frac{E_{S_n}(u(s)w(s)) - E_{S_n}(u(s))E_{S_n}(w(s))}{D_{S_n}(u(s))D_{S_n}(w(s))},$$

where $E_{S_n}()$ is the mean and $D_{S_n}()$ is the standard deviation of sample S_n.

It clearly follows that application of the correlation coefficient to global convexity measurement requires further adaptation, because the coefficient measures the strength of linear relationship between two variables $u(s)$ and $w(s)$ which are defined *for each object* of the sample, and any distance measure is a feature defined *for a pair of objects* from the sample: $d(s_i, s_j)$.

Jones and Forrest ([10]) avoided the problem of adaptation by using FDC only for problems with known global optima and measuring distance of a solution to the nearest one. Their distance measure was determined thus:

$$d_{s_{opt}}(s_i) = d(s_i, s_{opt})$$

(where s_{opt} is an optimum nearest to s_i). In this way they could easily compute FDC as $\hat{\rho}(d_{s_{opt}}, f)$ (f denotes the fitness function). But this approach requires knowledge about global optima, so it cannot be applied to practical problems.

In [2] Boese investigated not only distance to an optimum, but also average distance between a solution and all others in a sample:

$$\bar{d}(s_i) = \frac{1}{n-1} \sum_{j=1, j\neq i}^{n} d(s_i, s_j).$$

This method enabled him to find global convexity in instances of TSP without knowledge about global optima, so it is better suited for real-life applications.

Still, Boese's approach has its disadvantages. As it has been mentioned before, global convexity is a phenomenon characterised by smaller distance between better solutions. It does not specify, however, what happens in groups of solutions of poor quality (whether they are generally close to each other or not). So the operation of computing the average of distances between a solution and all others may cause some large distances to poor solutions to dominate the value of average $\bar{d}(s_i)$ and deteriorate the usability of $\hat{\rho}(\bar{d}, f)$ for measuring global convexity. This suggests the need for cautious averaging of $d(s_i, s_j)$ in a sample, with respect to quality of solutions.

An attempt to achieve this cautious averaging was made by Jaszkiewicz and Kominek in [9]. They introduced the notion of average distance from s_i to other solutions not worse than s_i. If a minimisation problem is considered and if solutions in sample S_n are sorted: $f(s_1) < f(s_2) < \ldots < f(s_n)$, then the average distance between a solution and solutions with not worse value of f may be defined as:

$$\hat{d}(s_i) = \frac{1}{i-1} \sum_{j=1}^{i-1} d(s_i, s_j) \qquad \text{for } i = 2, \ldots, n.$$

The above definition of distance is used in this paper in order to compute FDC as $\hat{\rho}(\hat{d}, f)$ in sample $S'_n = (s_2, \ldots, s_n)$ (s_1 is not used, since $\hat{d}(s_1)$ has not been defined).

6. The Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is, generally speaking, a problem of a transportation company who wants to define routes for its fleet of identical vehicles in order to deliver some goods from one central depot to customers ([11], [1]). Each customer may require different quantities of goods. Vehicles have certain maximum capacity constraint defined (maximum amount of goods that can be loaded). The company wants to minimise the total cost of deliveries.

More formally, let $G(V, E)$ be an undirected graph, $V = \{v_0, v_1, \ldots, v_N\}$, $N \geq 1$, is the set of vertices, v_0 represents the depot, while other vertices are customers.

Let c be the function of cost of an edge in G:

$$c\colon E \to \boldsymbol{R}_+ \cup \{0\}$$

and p be the customers' demand function:

$$p\colon V \setminus \{v_0\} \to \boldsymbol{R}_+$$

It is also assumed that:

$$\forall\ v \in V \setminus \{v_0\}\colon\ p(v) \leq Q,$$

where $Q > 0$ is the capacity constraint of a vehicle.

Under these assumptions the quadruple $I = (G, c, p, Q)$ is an instance of VRP.

A solution s of VRP is a set of $T(s)$ routes:

$$s = \{t_1, t_2, \ldots, t_{T(s)}\}.$$

A route has the form:

$$t_i = (v_0, v_{i,1}, v_{i,2}, \ldots, v_{i,n(t_i)}) \qquad \text{for } i = 1, \ldots, T(s),$$

where $n(t_i)$ is the number of customers on route t_i, the following constraints being satisfied:

$$\forall\ i \in \{1, \ldots, T(s)\}\ \forall\ k_i \in \{1, \ldots, n(t_i)\}\colon\ v_{i,k_i} \in V \setminus \{v_0\} \tag{1}$$

$$\forall\ i, j \in \{1, \ldots, T(s)\}\ \forall\ k_i \in \{1, \ldots, n(t_i)\},\ k_j \in \{1, \ldots, n(t_j)\}\colon$$
$$\big((i \neq j) \vee (k_i \neq k_j)\big) \Rightarrow (v_{i,k_i} \neq v_{j,k_j}) \tag{2}$$

$$\forall\ v \in V \setminus \{v_0\}\ \exists\ i \in \{1, \ldots, T(s)\},\ k_i \in \{1, \ldots, n(t_i)\}\colon\ v = v_{i,k_i} \tag{3}$$

$$\forall\ t_i \in s\colon\ \sum_{k_i=1}^{n(t_i)} p(v_{i,k_i}) \leq Q \tag{4}$$

Constraint 1 means that each vertex of each route (apart from v_0, the depot) represents some customer; constraints 2 and 3 require that each customer is visited exactly once; condition 4 assures that the sum of demands of customers on each route (serviced by one vehicle) does not exceed the capacity constraint.

Let S denote the set of all solutions of the form s. Let f denote the function of cost of a solution (cost of all edges traversed by vehicles):

$$f\colon S \to \boldsymbol{R}_+ \cup \{0\}$$

$$f(s) = \sum_{t_i \in s} \Big(c(v_0, v_{i,1}) + c(v_{i,n(t_i)}, v_0) + \sum_{k_i=1}^{n(t_i)-1} c(v_{i,k_i}, v_{i,k_i+1}) \Big)$$

The goal of VRP is to find such $s_{opt} \in S$ that:

$$\forall\ s \in S\colon\ f(s_{opt}) \leq f(s).$$

VRP is NP-hard ([12]) and even for relatively small instances with 75 customers global optima are not known. So in the past the problem was solved mainly by different heuristics ([4], [7]; see also [1]) and adapted metaheuristics, e.g. tabu search or evolutionary algorithms ([20], [21]).

However, none of the researchers who used evolutionary algorithms tried to systematically construct recombination operators for VRP; they used those suggested by intuition or adapted from other versions of VRP ([20], [19]). Thus, this paper describes the first attempt of such a construction based on global convexity tests.

Basic information about instances of VRP used in this work is given in table 1. These instances were taken from Taillard's website ([23]).

File name	Size (number of customers)	Total cost of the best-known solution
c100.dat	100	826.140
c100b.dat	100	819.560
c120.dat	120	1042.110
c150.dat	150	1028.420
c199.dat	199	1291.450
c50.dat	50	524.610
c75.dat	75	835.260
f134.dat	134	11629.600
f71.dat	71	241.970
tai100a.dat	100	2041.336
tai100b.dat	100	1940.610
tai100c.dat	100	1406.202
tai100d.dat	100	1581.250
tai150a.dat	150	3055.230
tai150b.dat	150	2727.770
tai150c.dat	150	2341.840
tai150d.dat	150	2645.390
tai75a.dat	75	1618.360
tai75b.dat	75	1344.640
tai75c.dat	75	1291.010
tai75d.dat	75	1365.420

Table 1. Basic informations about used instances.

The data files used include also information about the cost of the best-known solutions of VRP (cited in table 1) which comes mainly from [21]. Among the files are the very often used Christofides instances ([3]), with prefix 'c' in the name. Taillard's instances ([21]), with prefix 'tai', and Fischer's ones ([6]), with 'f', are also included.

7. Similarity measures for solutions of VRP

Although the notion of similarity measure is used instead of distance in this paper, all remarks made earlier (especially those concerning FDC) remain true, as similarity is just the opposite of distance.

7.1. Similarity as common pairs of nodes

First similarity measure takes into account only contents of routes of solutions regarded as sets, not as sequences. Assume that $s = \{t_1, \ldots, t_{T(s)}\}$. Its routes have the usual form: $t_i = (v_0, v_{i,1}, \ldots, v_{i,n(t_i)})$ for $i = 1, \ldots, T(s)$. Let's define the set of pairs of customers put together in a route:

$$PN(t_i) = \bigcup_{j=1}^{n(t_i)-1} \bigcup_{k=j+1}^{n(t_i)} \Big\{\{v_{i,j}, v_{i,k}\}\Big\}$$

It should be noticed that $PN(t)$ is empty if t connects only one customer with the depot; there are no pairs of customers put together in that route t.

The set of pairs of customers put together in some route of a solution is defined as:

$$PN(s) = \bigcup_{t_i \in s} PN(t_i)$$

For two solutions $s_1, s_2 \in S$ the union $PN(s_1) \cup PN(s_2)$ is the set of all pairs of customers put together in one route in at least one solution. The product $PN(s_1) \cap PN(s_2)$ consists of all pairs that are put together in some route in two solutions. Hence, the similarity of two solutions may be defined by the following formula:

$$sim_{cpn}(s_1, s_2) = \begin{cases} \sqrt{\frac{|PN(s_1) \cap PN(s_2)|}{|PN(s_1) \cup PN(s_2)|}} & \text{if } PN(s_1) \cup PN(s_2) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

The name, sim_{cpn}, means similarity in the sense of common pairs of nodes (customers).

The idea of this similarity measure comes from a two-stage heuristic strategy of solving VRP called 'cluster first – route second' ([7], [1]). This strategy is sustained by the intuition that in VRP it is important to have first good partition of customers into clusters (sets of customers in one route) and then to solve separately a set of TSP problems within these clusters.

7.2. Similarity as common edges

Second similarity measure regards routes as sets of edges. Let's define the multiset of edges of a route:

$$E(t_i) = \Big\{\{v_0, v_{i,1}\}\Big\} \cup \Big(\bigcup_{j=1}^{n(t_i)-1} \Big\{\{v_{i,j}, v_{i,j+1}\}\Big\}\Big) \cup \Big\{\{v_{i,n(t_i)}, v_0\}\Big\}$$

It is clearly stated that this is not a set in the strict meaning. Routes which connect only one customer with the depot have their only edge included twice. Because of this fact the operation of union is not simply the union of sets, but the union of multisets.

The multiset of edges of a solution is defined as:

$$E(s) = \bigcup_{t_i \in s} E(t_i)$$

(again, the symbol of union means the union of multisets).

Let's compute the cardinality of multiset $E(s)$; multisets of routes are all disjoint, so:

$$|E(s)| = \sum_{t_i \in s} |E(t_i)|$$

After simple counting of edges in the multiset of a route we have:

$$|E(t_i)| = n(t_i) + 1$$

thus:

$$|E(s)| = \sum_{t_i \in s} (n(t_i) + 1) = \sum_{i=1}^{T(s)} (n(t_i) + 1) = N + T(s)$$

It can be seen that solutions of one instance may vary with respect to the number of edges in their multisets E. Therefore the formula of similarity indicates what part of average size of multisets of edges is common:

$$sim_{ce}(s_1, s_2) = \frac{2 \cdot |E(s_1) \cap E(s_2)|}{N + T(s_1) + N + T(s_2)}$$

The name, sim_{ce}, means similarity in the sense of common edges.

The idea of this measure is taken from TSP. It is an adapted version of the measure used by Boese ([2]) to count common edges. It expresses the intuition that the presence of certain edges in solutions results in their good quality.

8. Global convexity tests

For each instance of VRP one sample of 1000 different local optima has been generated with the use of a randomized greedy local search procedure starting from random initial solutions. LS used two neighbourhood operators: exchange of two edges (2-opt) and connection of two routes (adapted from Clarke and Wright heuristic described in [4]); both of the neighbourhoods were checked in each iteration of the local search.

All computed values of FDC are negative. Their absolute values are shown in figure 2.

Negative values of FDC indicate the existence of a beneficial relationship between similarity and fitness: while similarity increases (solutions are closer to each other) the value of cost generally decreases (solutions have better quality). Moreover, absolute values of correlation for most of instances indicate that this relationship is rather strong; these values are below 0.6 only for instances f134 (for sim_{cpn}), c120 and tai75b (for sim_{ce}), and tai150a (for both of similarity measures). Hence, in only 5 cases of 42 the relationship is not strong.

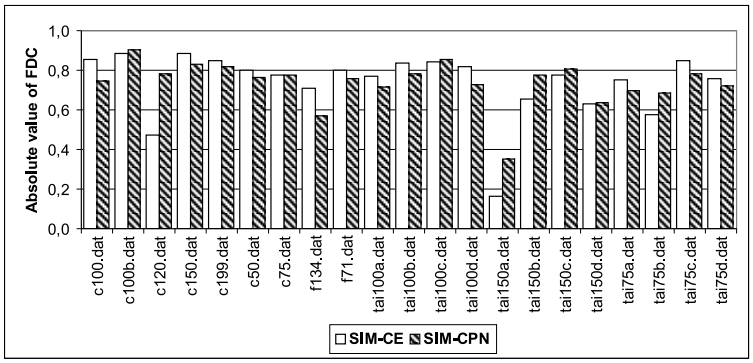

Figure 2. Absolute values of FDC for both of similarity measures and all instances.

Figures 3 and 4 show examples of scatter plots of fitness f and average similarity to not worse solutions (defined like $\hat{d}$). These figures are similar to those presented in [10], [2] and [9]. The instances chosen for these figures have the highest and the lowest absolute values of FDC (c100b and tai150a).

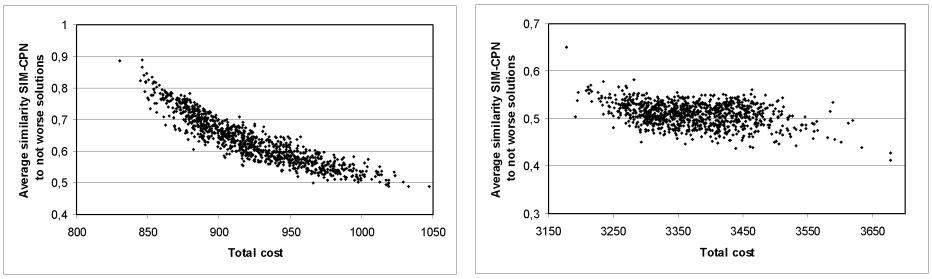

Figure 3. Scatter plots of fitness and $\hat{sim}_{cpn}$ (instances c100b and tai150a).

In the plots for instance c100b the existence of a 'path to the goal' ([10]) is clearly visible; there is a possibility of obtaining better solutions by making them more and more similar. This is exactly the picture of global convexity.

However, in the plots for instance tai150a the beneficial relationship is not visible. Hence, it may be stated that global convexity has not been revealed for this instance (with respect to measures sim_{cpn} and sim_{ce}).

To summarise, for most of instances of VRP strong global convexity has been revealed. Not only it allows to construct hopefully efficient distance preserving recombination operators, but also objectively confirms the intuition of many researchers regarding the nature of VRP, which is hidden in the definitions of similarity measures.

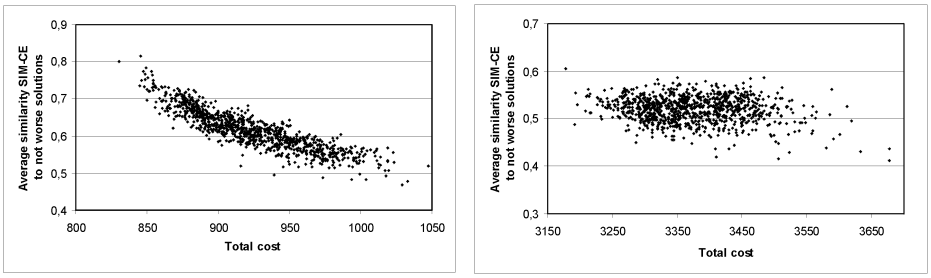

Figure 4. Scatter plots of fitness and $\hat{sim}_{ce}$ (instances c100b and tai150a).

9. Recombination operators

Three crossovers and one mutation have been constructed based on the definitions of measures sim_{cpn} and sim_{ce}. All these operators always result in feasible offsprings as no VRP constraint is violated during recombination.

In the descriptions of recombination operators below o denotes the offspring and p, p_1, p_2 represent parents.

9.1. CPX: Clusters Preserving Crossover

The general idea of this crossover operator is to preserve clusters of customers which are common to both of parents (as detected by measure sim_{cpn}).

Let's define:

$$Customers = V \setminus \{v_0\}$$

$$Clusters(s) = \bigcup_{t_i \in s} \Big\{ \{v_{i,1}, v_{i,2}, \ldots, v_{i,n(t_i)}\} \Big\}$$

The procedure of computing clusters which are common to solutions s_1 and s_2 may be as follows:

$CC = CommonClusters(s_1, s_2)$

$CC = \emptyset$
$C_1 = Clusters(s_1)$
$C_2 = Clusters(s_2)$
while ($C_1 \neq \{\emptyset\}$ and $C_2 \neq \{\emptyset\}$) do:
find $c_1 \in C_1$ and $c_2 \in C_2$ such that $sim_{cpn}(c_1, c_2)$ is maximum
if ($sim_{cpn}(c_1, c_2) > 0$) then $CC = CC \cup \{c_1 \cap c_2\}$
$C_1 = C_1 \setminus \{c_1\}$
$C_2 = C_2 \setminus \{c_2\}$

return CC

In this procedure, $sim_{cpn}(c_1, c_2)$ is the similarity of clusters defined in the same way as $sim_{cpn}(s_1, s_2)$. This measure is needed to determine clusters with largest intersections; only these parts are classified as common clusters.

The procedure of CPX:

$o = CPX(p_1, p_2)$

$o = \emptyset$
$C = Customers$
$CC = CommonClusters(p_1, p_2)$
for each $cc \in CC$ do $C = C \setminus cc$
for each $cc \in CC$ do $o = o \cup \{RandomRoute(cc)\}$
for each $v \in C$ do $o = o \cup \{(v_0, v)\}$
return o

$RandomRoute(cc)$ denotes a method of building a random route between customers in set (cluster) cc.

The above procedure simply builds random routes within clusters of customers common to both of parents. All customers which are not in these clusters are put in separate routes in the offspring.

9.2. CEPX: Common Edges Preserving Crossover

The idea of this crossover is to preserve all edges which are common to both of parents (as detected by measure sim_{ce}).

The folowing procedure defines CEPX:

$o = CEPX(p_1, p_2)$

$o = \emptyset$
$C = Customers$
$E_1 = E(p_1)$
$E_2 = E(p_2)$
remove from E_1 and E_2 all edges of the form $\{v_0, v\}$
$CE = E_1 \cap E_2$
for each $ce \in CE$ do $C = C \setminus ce$
$CP = AllMaximumPaths(CE)$
for each $cp \in CP$ do $o = o \cup \{Route(cp)\}$
for each $v \in C$ do $o = o \cup \{(v_0, v)\}$
return o

Since in set CE there are edges which may be connected through the same customers, $AllMaximumPaths(CE)$ is a method of computing all paths in the set. $Route(cp)$ is simply a route made from one path, e.g. if $cp = (v_1, v_2, v_3, v_4)$ (a path of 3 common edges) then $Route(cp) = (v_0, v_1, v_2, v_3, v_4)$.

The CEPX procedure creates routes out of paths common to both of parents. All customers which are not present on any common path are put in separate routes in the offspring.

9.3. CECPX: Common Edges and Clusters Preserving Crossover

Third crossover operator preserves both common edges and common clusters:

$o = CECPX(p_1, p_2)$

- $o = \emptyset$
- $C = Customers$
- $CC = CommonClusters(p_1, p_2)$
- for each $cc \in CC$ do $C = C \setminus cc$
- $E_1 = E(p_1)$
- $E_2 = E(p_2)$
- $CE = E_1 \cap E_2$
- $CP = AllMaximumPaths(CE)$
- for each $cc \in CC$ do:
  - $CCP = AllMaximumPathsInCluster(CE, cc)$
  - $CP = CP \setminus CCP$
  - $route = (v_0)$
  - make seq a random sequence of customers in cc and mark all customers as unused
  - while (seq contains any unused customers) do:
    - v is the first unused customer in seq
    - if (CCP contains a path $ccp = (\ldots, v, \ldots)$) then:
      - if (ccp starts or ends in v_0) then remove v_0 from the path
      - $route = route \cdot ccp$
      - mark all customers on path cp as used in seq
    - else:
      - $route = route \cdot (v)$
      - mark v as used in seq
  - $o = o \cup \{route\}$
- for each $cp \in CP$ do:
  - $o = o \cup \{Route(cp)\}$

remove all customers on cp from C

for each $v \in C$ do $o = o \cup \{(v_0, v)\}$

return o

$AllMaximumPathsInCluster(CE, cc)$ is a method of finding in set of edges CE all paths which consist only of customers from cluster cc (or v_0, the depot). Operation $route = route \cdot ccp$ is the concatenation of $route$ with path ccp.

The CECPX procedure first creates routes within common clusters using paths made from common edges. If there is no edge (path) connecting customers in a cluster, they are connected in a random sequence. Having created routes in clusters, the procedure creates routes from remaining paths and unclustered customers which are not present on any path (just like CEPX does).

9.4. CPM: Clusters Preserving Mutation

The idea of this mutation is to alter the contents of a parent in a way that does not change contents of any route perceived as a set of customers. The procedure is as follows:

$o = CPM(p)$

$o = \emptyset$

draw randomly one route t from p

copy all routes from p to o except t

cc is a set of customers on route t (a cluster)

$o = o \cup \{RandomRoute(cc)\}$

return o

The above procedure copies all routes from the parent to the offspring except one, which is changed in the random manner and then added to the offspring. This mutation creates such offsprings that always $sim_{cpn}(p, o) = 1$ (total similarity).

10. Computational experiments

In order to evaluate the recombination operators two different computational experiments have been conducted.

The first experiment compared GLS and MSLS algorithms in short runs with the same CPU time. MSLS (multiple-start local search) is a simple procedure that executes a LS in a given number of independent runs. The purpose of this experiment was to check whether new recombination operators ameliorate the computation process, when compared with simple local search.

The second experiment compared GLS algorithms with different operators in long runs in order to check speed and quality of these operators.

Three types of GLS have been used, each with one crossover operator and CPM as the mutation; the type of algorithm is indicated later by name: GLS-CPX, GLS-CEPX or GLS-CECPX. Other properties of these programs have been the same:

- the size of a population (K) set to 20;
- initial population consisted of one solution generated by Clarke and Wright heuristic ([4]), 10 solutions at most, generated by a modification of Gillet and Miller algorithm ([7]), and random solutions;
- randomized greedy local search procedure using the same neighbourhood operators as global convexity tests (2-opt and connection of two routes).

The MSLS program used the same LS procedure and the same initial solutions as GLS; having created all heuristic solutions, random ones were generated. GLS and MSLS shared the same pieces of code in order to assure fair comparison.

Quality of solutions has been measured by the following formula:

$$quality(s) = \frac{f(s) - f(best\_known)}{f(best\_known)} \cdot 100\ [\%],$$

where $f(best\_known)$ is the cost of the best known solution for an instance, as shown in table 1.

10.1. Experimental setup 1: GLS bound by MSLS

In this experiment 10 sets of 1000 solutions have been generated by MSLS first. The average time of one run (1000 solutions) has been used as the time limit for GLS. Then each type of GLS has been run 10 times. The programs have been executed on a PC with Intel Celeron 900 MHz processor running under Windows 2000.

The quality of the best solutions found in all 10 runs by each algorithm is shown in figure 5, whereas figure 6 shows averages of the best solutions found in all runs.

In both of the figures it can be seen that for every instance GLS is better than MSLS. The gains in quality are highest for the biggest instances, because smaller ones are already solved almost to the best-known values by MSLS.

Among memetic algorithms GLS-CPX generates the worst solutions; GLS-CEPX and GLS-CECPX are rather indistinguishable from each other with respect to quality.

The general conclusion is that recombination operators indeed ameliorate the computation process in a very significant way; preservation of distance by these operators results in better solutions.

10.2. Experimental setup 2: unbound GLS

In this experiment each type of GLS has been run with the same stopping condition: 200 iterations withouth amelioration of the best solution in the population. The condition was set so in order to allow GLS to converge, no matter how much time it takes. Each GLS has

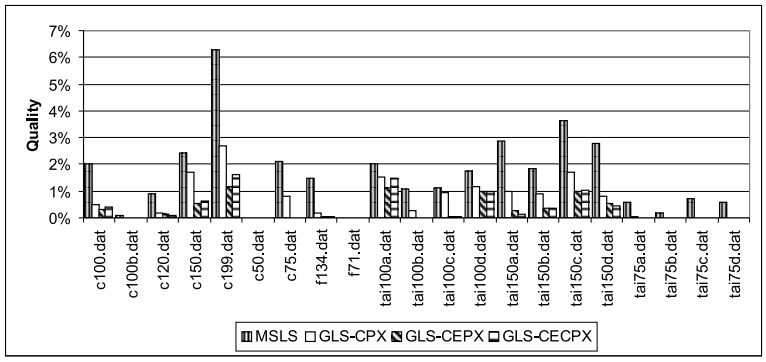

Figure 5. Quality of the best solutions of all 10 runs for all instances.

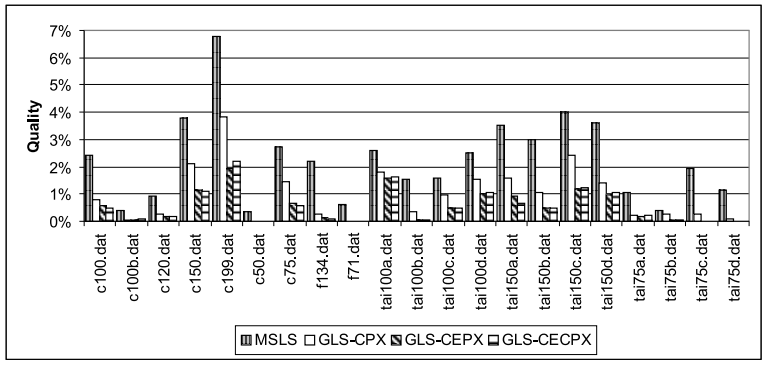

Figure 6. Average quality of the best solutions in 10 runs for all instances.

been given 10 such trials. The programs have been executed on a PC with AMD Duron 1200 processor running under Windows 2000 (it resulted in a computation 3-3.5 times faster than in the previous setup).

In figure 7 the quality of the best solutions found in all 10 runs by each algorithm is shown. Figure 8 shows averages of the best solutions found in all runs.

From both of the figures it can be seen that the GLS using CPX operator is the worst. Although it finds the best solutions in 8 cases (including the cases which ended in a draw), GLS-CEPX is the best in 12 cases and GLS-CECPX in 14 cases. Furthermore, also a look at the average quality of the best solutions in all 10 runs indicates that the version with CPX operator is the worst, while GLS-CEPX and GLS-CECPX are rather indistinguishable (the latter being perhaps slightly better).

Figure 9 shows time of computation of each type of GLS (curves of second order polynomial regression).

The values of R^2 are very high (above 0.97), which means that these curves show well

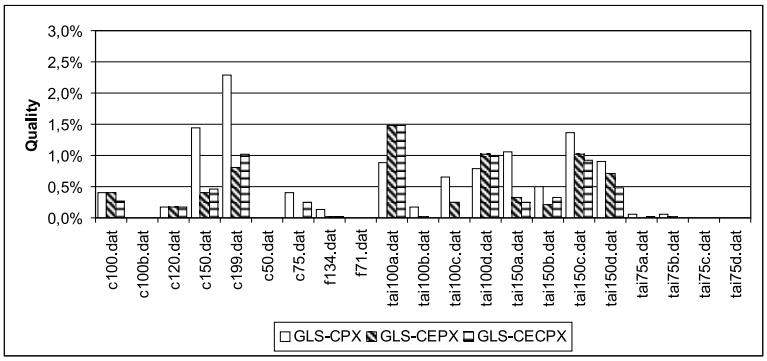

Figure 7. Quality of the best solutions of all 10 runs for all instances.

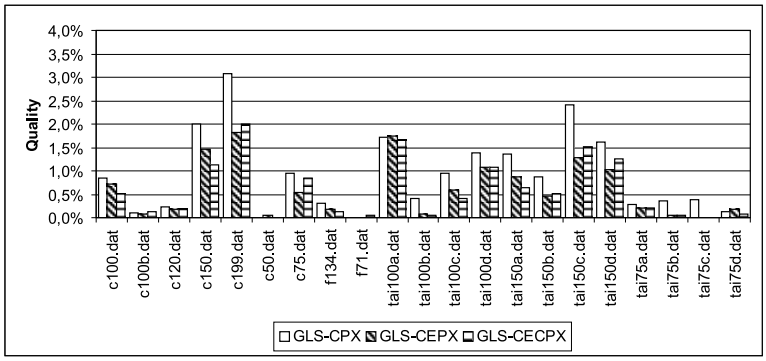

Figure 8. Average quality of the best solutions in 10 runs for all instances.

the computation time of each type of GLS. These curves indicate that GLS-CPX is not only of the worst quality, but also the most time-consuming algorithm. Of the remaining two, the faster one is GLS-CECPX. Hence, also in case of this experiment it may be concluded that preservation of both of important features of solutions is very beneficial: it increases the speed of GLS without any deterioration in quality of solutions.

11. Conclusions and further work

One of the most important results of the work presented in this paper is that strong global convexity has been revealed in almost all used instances of VRP. Also, global convexity in landscapes formed by sim_{cpn} measure objectively confirms the intuition of the approach called 'cluster first – route second' which was used in some heuristic algorithms.

The results of global convexity tests and the definition of similarity measures allowed for

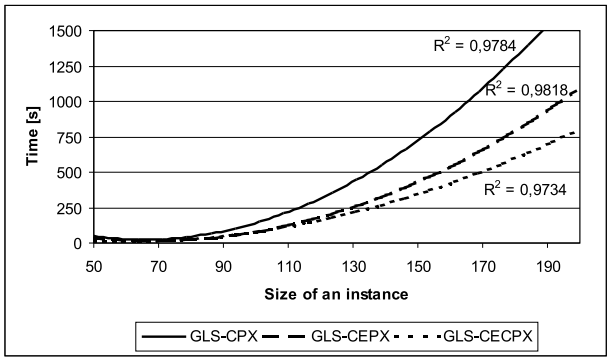

Figure 9. Average time of computation of GLS (curves of regression).

construction of distance preserving recombination operators (3 crossovers and a mutation) which, in the scheme of the genetic local search, indeed ameliorate the search process in comparison to simple local search and generate very good solutions of VRP.

Also in the particular case of VRP it has been shown that preservation of both of important features of solutions (common edges and clusters) in CECPX operator resulted in significantly faster computation without loss in quality. Perhaps a hypothesis may be stated that the more important features of solutions are preserved in a DPX, the better an algorithm is.

This paper is, therefore, another evidence that systematic construction of recombination operators is sensible in case of the genetic local search. It also confirms that there is a need for systematic and justified adaptation of metaheuristics to problems and that the means of this adaptation (such as global convexity tests) should be investigated more closely.

Further research will firstly check the definition and implementation of crossover operators. In author's opinion there is a possibility of an important change in $CommonClusters$ procedure, which most probably does not compute all possible common parts of clusters in two solutions. The change should lead to an even more efficient implementation of CPX and CECPX operators.

Also, a modification of the search space of the local search procedure in GLS by forbidding certain movements (as described by Jaszkiewicz in [8]) seems to be very promising and might be implemented in order to accelerate computation.

The next, very important step needed for a proper evaluation of the recombination operators described here will be to conduct a comparison experiment with other operators for VRP described in literature, such as SBX, RBX ([19]), and SPLIT-crossover ([20]).

Acknowledgements

I would like to thank Dr. Andrzej Jaszkiewicz, whose inspiration and suggestions enriched this work, and Prof. Roman Słowiński, whose encouragement helped to complete this paper.

References

[1] Aronson L. D., Algorithms for Vehicle Routing – a Survey, Report 96-21, Delft University of Technology, 1996.

[2] Boese K. D., Cost Versus Distance in the Traveling Salesman Problem, Tech. Rep. TR-950018, UCLA CS Departament, 1995.

[3] Christofides N., et al. (eds.), *Combinatorial Optimization*, John Wiley, Chichester, 1979.

[4] Clarke G., Wright J., Scheduling of Vehicles from a Central Depot to a Number of Delivery Points, *Operations Research*, **12**, 1964, 568-582.

[5] Davis L., Adapting Operator Probabilities in Genetic Algorithms, in: J. Schaffer (ed.), *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*, Morgan Kaufmann, San Mateo, CA, 1989, 61-69.

[6] Fisher M. L., Optimal solution of vehicle routing problems using minimum K-trees, *Operations Research*, **42**, 1994, 626-642.

[7] Gillet B. E., Miller L. R., A Heuristic Algorithm for the Vehicle Dispatch Problem, *Operations Research*, **22**, 1974, 340-349.

[8] Jaszkiewicz A., Improving performance of genetic local search by changing local search space topology, *Foundations of Computing and Decision Sciences*, **24**, 2, 1999.

[9] Jaszkiewicz A., Kominek P., Genetic Local Search with Distance Preserving Recombination Operator for a Vehicle Routing Problem, *European Journal of Operational Research*, 2003, in preparation.

[10] Jones T., Forrest S., Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms, in: L. J. Eshelman (ed.), *Proceedings of the 6th Int. Conference on Genetic Algorithms*, Kaufmann, 1995, 184-192.

[11] Lawler E. L., et al. (eds.), *The Traveling Salesman Problem, A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, 1985.

[12] Lenstra J. K., Rinnooy Kan A. H. G., Complexity of vehicle routing and scheduling problem, *Networks*, **11**, 1981, 221-227.

[13] Merz P., Freisleben B., Fitness Landscapes and Memetic Algorithms Design, in: D. Corne, et al. (eds.), *New Ideas in Optimization*, McGraw–Hill, 1999.

[14] Merz P., Freisleben B., Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem, *IEEE Transactions on Evolutionary Computation*, **4**, 4, 2000, 337-352.

[15] Merz P., Freisleben B., New Genetic Local Search Operators for the Traveling Salesman Problem, in: H.-M. Voigt, et al. (eds.), *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature – PPSN IV*, Springer, 1996, 890-900.

[16] Merz P., Freisleben B., Genetic Local Search for the TSP: New Results, in: *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, IEEE Press, 1997, 159-164.

[17] Michalewicz Z., A Hierarchy of Evolution Programs: An Experimental Study, *Evolutionary Computation*, **1**, 1, 1993, 51-76.

[18] Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer–Verlag, Berlin Heidelberg, 1992.

[19] Potvin J.-Y., Bengio S., The vehicle routing problem with time windows part ii: genetic search, *INFORMS Journal of Computing*, **8**, 2, 1996.

[20] Prins C., A Simple and Effective Evolutionary Algorithm for the Vehicle Routing Problem, *MIC'2001 – 4th Metaheuristics International Conference*, July 2001.

[21] Rochat Y., Taillard É. D., Probabilistic Diversification and Intensification in Local Search for Vehicle Routing, *Journal of Heuristics*, **1**, 1995, 147-167.

[22] Stadler P. F., Landscapes and their Correlation Functions, *J. Math. Chem.*, **20**, 1996, 1-45.

[23] Taillard É. D., website with instances of VRP, http://www.eivd.ch/ina/Collaborateurs/etd/problemes.dir/vrp.dir/vrp.html.

[24] Weinberger E. D., Correlated and Uncorrelated Landscapes and How to Tell the Difference, *Biological Cybernetics*, **63**, 1990, 325-336.

[25] Wolpert D. H., Macready W. G., No Free Lunch Theorem for Optimization, *IEEE Transactions on Evolutionary Computation*, **1**, 1, 1997, 67-82.