Experimental Evaluation of Two Approaches to Optimal Recombination for Permutation Problems*

Anton V. Eremeev[1] and Julia V. Kovalenko[2]

May 4, 2016

[1] Omsk Branch of Sobolev Institute of Mathematics,
13 Pevtsov str. 644043, Omsk, Russia.
email: eremeev@ofim.oscsbras.ru,

[2] Sobolev Institute of Mathematics,
4, Akad. Koptyug avenue, 630090, Novosibirsk, Russia.
email: julia.kovalenko.ya@yandex.ru

Abstract

We consider two approaches to formulation and solving of optimal recombination problems arising as supplementary problems in genetic algorithms for the Asymmetric Travelling Salesman Problem and the Makespan Minimization Problem on a Single Machine. All four optimal recombination problems under consideration are NP-hard but relatively fast exponential-time algorithms are known for solving them. The experimental evaluation carried out in this paper shows that the two approaches to optimal recombination are competitive with each other.

Keywords: genetic algorithm, optimal recombination problem, permutation

Introduction

Performance of genetic algorithms (GA) depends significantly upon the choice of the crossover operator, where the components of parent solutions are combined to build the offspring. *Optimal recombination problem* (ORP) consists in finding the best possible offspring as a result of a crossover operator, given two feasible parent solutions. The ORP is a supplementary problem (usually) of smaller dimension than the original problem, formulated in view of the basic principles of crossover [1]. The experimental results of M. Yagiura and T. Ibaraki [2], C. Cotta, E. Alba and J.M. Troya [3], W. Cook and

*The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-30698-8_10].

P. Seymour [4], R. Tinós, D. Whitley and G. Ochoa [5] indicate that optimal recombination may be used successfully in the genetic algorithms for problems on permutations.

This paper is devoted to analysis and comparison of two approaches to construct the optimal recombination operators for Asymmetric Travelling Salesman Problem (ATSP) and Makespan Minimization Problem on a Single Machine ($1|s_{vu}|C_{\max}$).

The first approach is based on the preservation of elements in positions of the parent permutations. Following this approach for ATSP and $1|s_{vu}|C_{\max}$ we show that on one hand the ORP with position-based representation is strongly NP-hard, on the other hand, almost all of the ORP instances are efficiently solvable. We develop an exact algorithm for solving the ORP, using enumeration of all possible combinations of the maximal matchings in cycles of a special bipartite graph. A crossover operator, where this ORP is solved by means of the proposed algorithm, is called here *Optimized Cycle Crossover* (OCX) and may be considered as a derandomization of Uniform Cycle Crossover [6].

The second approach is based on the preservation of the adjacencies found in the parents. Strong NP-hardness of the ORP with adjacency-based representation is proven and a solution method is proposed. A crossover operator, which solves this ORP, is called here *Optimized Directed Edge Crossover* (ODEC) and may be considered as a "direct descendant" of Directed Edge Crossover [7].

The theoretical worst-case and average-case upper bounds on time complexity of optimized crossovers obtained in this paper and in [14] are not sufficient to estimate efficiency of GAs based on such operators. Even if the time complexity of one optimized crossover is greater than the other, this does not necessarily mean that a GA using the slower crossover will require more time to find an optimal solution. Note that given the same pair of parent solutions, the slower crossover operator may be choosing the best offspring from a larger set of possible offspring solutions, thus giving more advantage to the GA. Therefore, a deeper analysis is required in order to decide which of the two approaches is the most appropriate for a given problem. We perform computational experiments to compare the behavior of GAs that use the optimized crossovers based on the two approaches mentioned above.

A wide spectrum of metaheuristics and heuristics has been proposed to the ATSP problem (see e.g. [8, 9, 10]). Many of these algorithms solve the ATSP instances from [11] very quickly. Note, however, that the present paper is aimed, first of all, at comparison of the optimized crossovers, rather than constructing fast algorithms for ATSP problem. In particular, in the computational experiments we test the optimized crossovers in a very basic GA with elitist recombination without any problem-specific local search procedures or fine tuning of parameters.

The paper is organized as follows. In Section 1, we provide a formal description of the Optimal Recombination Problem. Section 2 is devoted to the theoretical analysis of computational complexity of the two ORPs for Makespan Minimization Problem on a Single Machine. A similar analysis for Asymmetric Travelling Salesman Problem is provided in Section 3. The computational experiments are described in Section 4 and the

concluding remarks are given in Section 5.

1 Optimal Recombination Problem

The genetic algorithm is a random search method that models a process of evolution of a population of *individuals* [12]. Each individual is a sample solution to the optimization problem being solved. For the sake of generality in this section we can consider combinatorial optimization problems, where the solutions are represented by strings of length n, composed of symbols from some finite alphabet. The components of these stings are called *genes*. Individuals of a new population are built by means of variation operators (crossover and/or mutation).

Performance of the GA depends significantly upon the choice of the crossover operator, where genes of parent individuals are combined to build the offspring. Optimal Recombination Problem consists in finding the best possible offspring as a result of a crossover operator, given two parent individuals. The following definition of the optimal recombination problem is motivated by the principles of (strictly) gene transmitting recombination formulated by N. Radcliffe [1].

Given: an instance I of combinatorial optimization problem with the set of feasible solutions Sol, objective function f : Sol $\rightarrow$ R, and two parent solutions $\mathbf{x}^1 = (x_1^1, \ldots, x_n^1), \mathbf{x}^2 = (x_1^2, \ldots, x_n^2)$ from Sol.
Find: a feasible solution (offspring) $\mathbf{x}' = (x_1', \ldots, x_n')$ such that

(i) $x_j' = x_j^1$ or $x_j' = x_j^2$ for all $j = 1, \ldots, n$;

(ii) for each $\mathbf{x} \in$ Sol such that $x_j = x_j^1$ or $x_j = x_j^2$, $j = 1, \ldots, n$, the inequality

$$f(\mathbf{x}') \leq f(\mathbf{x}) \text{ holds in the case of minimization problem,}$$

or

$$f(\mathbf{x}') \geq f(\mathbf{x}) \text{ holds in the case of maximization problem.}$$

Note that in the case of permutation problems, the set of feasible solutions Sol consists of permutations, so the offspring is required to be a permutation too.

2 Makespan Minimization Problem on a Single Machine

Consider the Makespan Minimization Problem on a Single Machine $(1|s_{vu}|C_{\max})$, which is equivalent to the problem of finding the shortest Hamiltonian path in a digraph.

The input consists of a set of jobs $V = \{v_1, \ldots, v_k\}$ with positive processing times d_v, $v \in V$. All jobs are available for processing at time zero, and preemption is not allowed. A sequence dependent setup time is required to switch a machine from one job to another. Let s_{vu} be the a non-negative setup time from job v to job u for all $v, u \in V$, where $v \neq u$. The goal is to schedule the jobs on a single machine so as to minimize the maximum job completion time, the so-called *makespan* $C_{\max}$.

Problem $1|s_{vu}|C_{\max}$ is strongly NP-hard [13], and cannot be approximated with any constant or polynomial factor of the optimum in polynomial time, unless P=NP. Therefore metaheuristics, in particular, genetic algorithms, are appropriate for this problem.

The feasible solutions of $1|s_{vu}|C_{\max}$ can be represented in two natural ways in a GA: (I) genes encode jobs and (II) genes encode adjacencies. Let us consider these representations and the ORPs that correspond to them.

2.1 Optimal Recombination with Position-Based Representation

Let $\pi = (\pi_1, \ldots, \pi_k)$ denote a permutation of the jobs, i.e. π_i is the i-th job on the machine, $i = 1, \ldots, k$. Put $s(\pi) = \sum_{i=1}^{k-1} s_{\pi_i, \pi_{i+1}}$. Then the problem $1|s_{vu}|C_{\max}$ is equivalent to finding a permutation π^* that minimizes the total setup time $s(\pi^*)$.

The ORP for problem $1|s_{vu}|C_{\max}$ with position-based representation, given two parent solutions π^1 and π^2, asks for a permutation π' such that:

(i) $\pi'_i = \pi^1_i$ or $\pi'_i = \pi^2_i$ for all $i = 1, \ldots, k$;

(ii) π' has the minimum value of objective function $s(\pi')$ among all permutations that satisfy condition (i).

The following theorem is obtained in [14].

Theorem 1 *The ORP for problem* $1|s_{vu}|C_{\max}$ *with position-based representation is strongly NP-hard.*

We build an algorithm for solving the formulated ORP, using the approach of A.I. Serdyukov [15] which was developed for solving the travelling salesman problem with vertex requisitions.

Let us consider a bipartite graph $G = (V_n, V, U)$ where the two subsets of vertices of bipartition $V_n = \{1, \ldots, n\}$ and V have equal sizes and the set of edges is $U = \{\{i, v\} : i \in V_n,\ v = \pi^1_i \text{ or } v = \pi^2_i\}$. Now there is a one-to-one correspondence between the set of perfect matchings in graph G and the set of feasible solutions to an ORP instance with parents $\pi^1,\ \pi^2$: Given a perfect matching of the form $\{\{1, v^1\}, \{2, v^2\}, \ldots, \{n, v^n\}\}$, this mapping produces the permutation of jobs $(v^1, v^2, \ldots, v^n)$.

An edge $\{i, v\} \in U$ is called *special*, if $\{i, v\}$ belongs to all perfect matchings in graph G. Note that a maximal (by inclusion) connected subgraph of graph G with at least two edges is a *cycle*. Let $q(G)$ denote the number of cycles in graph G. The edges $\{i, v\} \in U$, such that $\pi_i^1 = \pi_i^2$, are special and belong to none of the cycles, while the edges $\{i, v\} \in U$, such that $\pi_i^1 \neq \pi_i^2$, belong to some cycles. Besides that, each cycle $j,\ j = 1, \ldots, q(G)$, of graph G contains exactly two maximal (edge disjoint) matchings, so it does not contain the special edges. Hence an edge $\{i, v\} \in U$ is special iff $\pi_i^1 = \pi_i^2$, and every perfect matching in G is defined by a combination of maximal matchings chosen in each of the cycles and the set of all special edges.

The cycles of graph G may be computed in $O(k)$ time, e.g. by means of the "depth first" algorithm [16]. The special edges and maximal matchings in cycles may be found easily in $O(k)$ time.

Therefore, the ORP with position-based representation is solvable by the following algorithm: Build the bipartite graph G, identify the set of special edges and cycles and find all maximal matchings in cycles. Enumerate all perfect matchings of graph G by combining the maximal matchings of cycles and joining them with special edges. During enumeration, each of the perfect matchings is assigned the corresponding permutation π of jobs and $s(\pi)$ is computed. As a result, one can find the required permutation π'.

The total number of perfect matchings in graph G is equal to $2^{q(G)}$, so the time complexity of the above algorithm is $O(k2^{q(G)})$, where $q(G) \leq \lfloor \frac{k}{2} \rfloor$ and this bound is tight. Note that the proposed algorithm can be used for different objective functions defined on the set of permutations (see examples in [2, 3, 6, 17]).

In [14], a modification of the described algorithm was proposed to speed up the evaluation of makespan function in the process of perfect matching enumeration. This modification performs a preprocessing stage, where the values of makespan function for *cycle contacts* are computed, and solves the ORP for $1|s_{vu}|C_{\max}$ in $O\left(q(G)2^{q(G)} + q(G)k\right)$ time.

Moreover in [14], it was shown that for almost all pairs of parent solutions $q(G) \leq \frac{\ln(k)}{\ln(2)}$, i.e. the cardinality of the set of feasible solutions is at most k. To describe this result precisely, let us give the following

Definition 1 [15] *A graph $G = (V_k, V, U)$ is called "good" if it satisfies the inequality $q(G) \leq \frac{\ln(k)}{\ln(2)}$.*

Let $\bar{\Re}_k$ denote the set of pairs of parent solutions with k jobs which correspond to "good" bipartite graphs G and let $\Re_k$ be the set of all pairs of parent solutions with k jobs. The results from [14] imply

Theorem 2 $|\bar{\Re}_k|/|\Re_k| \longrightarrow 1$ *as* $k \to \infty$.

According to the frequently used terminology (see e.g. [18]), this theorem means that *almost all* of the ORP instances have at most k feasible solutions and thus solvable in $O(k\ln(k))$ time.

In what follows, the crossover operator, solving the ORP by means of the algorithm described above, will be called Optimized Cycle Crossover. Such crossover may be considered as a derandomization of Uniform Cycle Crossover, which constructs an offspring so that maximal matching is chosen randomly in each of the cycles [6].

2.2 Optimal Recombination with Adjacency-Based Representation

Consider representation of solutions based on adjacencies. Here a solution is encoded as a vector $\mathbf{p} = (p_1, \ldots, p_k)$, where p_i is the job that immediately precedes job $v_i,\ i = 1, \ldots, k$. We assume that $p_i = v_0$ marks the first element of the sequence, where v_0 is an *artificial* job and $s_{v_0 v} = 0$ for all $v \in V$. Then $s(\mathbf{p}) = \sum_{i=1}^{k} s_{p_i, v_i}$. The problem $1|s_{vu}|C_{\max}$ is equivalent to finding a permutation $\mathbf{p}^*$ that minimizes the total setup time $s(\mathbf{p}^*)$.

The ORP for problem $1|s_{vu}|C_{\max}$ with adjacency-based representation, given two parent solutions $\mathbf{p}^1$ and $\mathbf{p}^2$, asks for a feasible solution $\mathbf{p}'$ such that:
(i) $p'_i = p^1_i$ or $p'_i = p^2_i$ for all $i = 1, \ldots, k$;
(ii) $\mathbf{p}'$ has the minimum value of objective function $s(\mathbf{p}')$ among all solutions that satisfy condition (i).

Theorem 3 *The ORP for problem* $1|s_{vu}|C_{\max}$ *with adjacency-based representation is strongly NP-hard.*

The proof is based on the known result from [14] about NP-hardness of the ORP for ATSP with adjacency-based representation (see Section 3). In the proof of NP-hardness of this ORP in Theorem 1.3 in [14], the vertex cover problem is reduced to it in such a way that there are arcs belonging to both parent tours (and thus should belong to the offspring tour). Let us take one of these arcs $(v_\ell, v_{\ell'})$ and delete it from both of the parent tours. The remaining two Hamilton paths will be used now to build a pair of parents for the $1|s_{vu}|C_{\max}$ ORP.

Suppose that an instance of ATSP (an n-vertex graph and arc weights c_{ij}, $i = 1, \ldots, n,\ j = 1, \ldots, n$) is given as defined in the proof of Theorem 1.3 from [14]. Let us construct an instance of $1|s_{vu}|C_{\max}$ problem with $k = n + 1$, where $s_{v_i, v_j} = c_{ij}$ for all $i = 1, \ldots, n,\ j = 1, \ldots, n$. The job v_{n+1} is introduced to ensure that the offspring solution will end with job v_ℓ. The setup times associated with v_{n+1} are set to zero, i.e. $s_{v_i, v_{n+1}} = s_{v_{n+1}, v_i} = 0$. Suppose that two tours $v_{\ell'} = v_{i_1}, v_{i_2}, \ldots, v_{i_n} = v_\ell, v_{\ell'}$ and $v_{\ell'} = v_{j_1}, v_{j_2} \ldots, v_{j_n} = v_\ell, v_{\ell'}$ are the parent solutions of the ORP instance for ATSP constructed in the proof of Theorem 1.3 from [14]. Then the two parent solutions $\mathbf{p}^1$ and $\mathbf{p}^2$ for the $1|s_{vu}|C_{\max}$ ORP problem with adjacency-based representation are defined as follows: $p^1_{i_1} = v_0,\ p^1_{i_2} = v_{i_1}, \ldots, p^1_{i_n} = v_{i_{n-1}},\ p^1_{n+1} = v_{i_n}$ and $p^2_{j_1} = v_0,\ p^2_{j_2} = v_{j_1}, \ldots, p^2_{j_n} = v_{j_{n-1}},\ p^2_{n+1} = v_{j_n}$. The first and the last setups are the

same in these schedules so an optimal ORP solution to $1|s_{vu}|C_{\max}$ will define an optimal ORP solution for ATSP, which is NP-hard. The described transformations are efficiently computable, so the ORP for $1|s_{vu}|C_{\max}$ problem is NP-hard as well. Q.E.D.

The optimized crossover operator, which solves the ORP for $1|s_{vu}|C_{\max}$ with adjacency-based representation, will be called Optimized Directed Edge Crossover and may be considered as a deterministic "direct descendant" of Directed Edge Crossover [7]. However, unlike the latter one, Optimized Directed Edge Crossover guarantees a gene transmitting recombination.

The following theorem gives an upper bound on time-complexity of ODEC.

Theorem 4 *The ORP for problem* $1|s_{vu}|C_{\max}$ *with adjacency-based representation is solvable in* $O(k^2 2^{\frac{k}{2}})$ *time.*

The proof is based on a Turing reduction of the ORP to $O(k)$ instances of Travelling Salesman Problem with forced edges on cubic graphs, i.e. the graphs with maximal degree three. Let us consider the ORP for $1|s_{vu}|C_{\max}$ with parent solutions $\mathbf{p}^1$ and $\mathbf{p}^2$ as a Shortest Hamilton Path problem on $(k+1)$-vertex digraph $G' = (V \cup \{v_0\}, A)$ where the arcs correspond to setups presented in $\mathbf{p}^1$ and $\mathbf{p}^2$ and the arc costs c_{ij} are equal to the setup times $s_{v_i v_j},\ i = 0, 1, \ldots, k, j = 1, \ldots, k$. Add a zero-cost arc (v, v_0) where $v \in V$ (enumerate all $O(k)$ options to choose $v \in V$). The resulting digraph is denoted by G'_v. This digraph may be transformed into a cubic graph $\overline{G}_v$ with forced edges the same way as in the ORP for ATSP [14]. The graph $\overline{G}_v$ is constructed so that the setups presented in both parent solutions $\mathbf{p}^1$ and $\mathbf{p}^2$ correspond to forced edges.

All Hamiltonian cycles in $\overline{G}_v$ w.r.t. the set of forced edges are enumerated in time $O(2^{\frac{k}{2}})$ by the algorithm of D. Eppstein [19]. Then, for each Hamiltonian cycle C from $\overline{G}_v$ in each of the two directions we can check if it is possible to pass a circuit in G'_v through the arcs corresponding to edges of C, and if possible, compute the cost of the circuit. So, the ATSP problem on graph G'_v is solvable in $O(k2^{\frac{k}{2}})$ time and, therefore, solving the ATSP problems on graphs G'_v for all $v \in V$ requires $O(k^2 2^{\frac{k}{2}})$ time. Q.E.D.

3 Asymmetric Travelling Salesman Problem

In this section, we briefly consider the Travelling Salesman Problem (TSP). Suppose a complete digraph $\bar{G}$ is given. The set of vertices of $\bar{G}$ is $V = \{v_1, \ldots, v_n\}$ and a set of arcs is $A = \{(v_i, v_j) :\ v_i, v_j \in V,\ i \neq j\}$. A weight (length) $c_{ij} \geq 0$ of each arc $(v_i, v_j) \in A$ is given as well. It is required to find a Hamiltonian circuit of minimum length. If $c_{ij} \neq c_{ji}$ for at least one (v_i, v_j) then the TSP is called Asymmetric Travelling Salesman Problem (ATSP).

Feasible solution to the ATSP may be encoded as a sequence of the vertices in the TSP tour (without loss of generality we assume that the first position contains vertex v_1), or as a vector of adjacencies, where the immediate predecessor is indicated for each vertex.

Position-Based Representation. In the case of position-based encoding of solutions in ATSP, the ORP may be solved by the means of the algorithm described in Subsection 2.1. A slight modification of the speed-up method from [14] is applicable here as well. Therefore, the ORP for ATSP with position-based representation is solvable in $O(n2^{\frac{n}{2}})$ time and almost all of its instances are solvable in $O(n\ln(n))$ time.

The following theorem is proved analogously to Theorem 2.2 form [14].

Theorem 5 *The ORP for ATSP with position-based representation is strongly NP-hard.*

Adjacency-Based Representation. The ORP for the ATSP with adjacency-based representation is shown to be strongly NP-hard but solvable in $O(n2^{\frac{n}{2}})$ time [14].

4 Computational Experiment on TSPLIB Instances

4.1 Genetic Algorithm

M. Yagiura, T. Ibaraki [2] applied a genetic algorithm with *elitist recombination* [20] to a number of combinatorial optimization problems on permutations. Let us consider the scheme of the GA with elitist recombination in a general form as it may be applied to a combinatorial minimization problem from Section 1.

Genetic Algorithm with Elitist Recombination

STEP 1. Construct the initial population.

STEP 2. Assign $t := 1$.

STEP 3. Repeat steps 3.1-3.4 until some stopping criterion is satisfied:

3.1. Choose randomly two parent individuals $\mathbf{x}^1, \mathbf{x}^2$ from the population.

3.2. Create an offspring $\mathbf{x}'$, applying a crossover to $\mathbf{x}^1$ and $\mathbf{x}^2$.

3.3. Replace one of the two parents by $\mathbf{x}'$.

3.4. $t := t + 1$.

STEP 4. The result is the best found individual w.r.t. objective function.

In our implementation of the GA with elitist recombination the *arbitrary insertion* method [2] is used for generating individuals of the initial population on Step 1. The population size N remains constant during the execution of the GA.

We apply an optimized crossover (ODEC or OCX) to generate a new individual on Step 3.2. One of the parents $\mathbf{x}^1$, $\mathbf{x}^2$ is replaced by the offspring as follows. We suppose without loss of generality that $f(\mathbf{x}^1) \leq f(\mathbf{x}^2)$. Replace $\mathbf{x}^2$ by $\mathbf{x}'$ with probability $P(\Delta_1/\Delta_2)$, otherwise replace $\mathbf{x}^1$ by $\mathbf{x}'$, where

$$\Delta_i = f(\mathbf{x}^i) - f(\mathbf{x}'),\ i = 1, 2, \tag{1}$$

$$P(\Delta_1/\Delta_2) = \min\Big\{\frac{\Delta_1/\Delta_2}{a}, 1\Big\}. \tag{2}$$

Note that $\Delta_2 \geq \Delta_1 \geq 0$ by the definition, and hence $\Delta_1/\Delta_2 \in [0,1]$ (we consider $\Delta_1/\Delta_2 = 1$ if $\Delta_1 = \Delta_2 = 0$). The constant $a \geq 0$ is a tunable parameter. If $a = 0$, then p' always replaces p^2, and if $a = \infty$, then p' always replaces p^1.

The described GA was programmed in Java (NetBeans IDE 7.2.1) and tested on a computer with Intel Core 2 Duo CPU E7200 2,53GHz processor, 2 Gb RAM.

4.2 Testing Problems and Experimental Outline

In the computational experiment, the described above GA was applied to ATSP and $1|s_{vu}|C_{\max}$ problems for evaluation of the effects of different optimized crossovers. Population size N was set to 50 and the tunable parameter a was set to 0.5.

In the experiments, we used the ATSP instances from TSPLIB [11] library. The ATSP collection includes instances from different applications [21, 22, 23]. The rbg instances come from a stacker crane application. The two ft instances arise in a problem of optimal sequencing tasks in the coloring plant of a resin production department. The ftv instances are from vehicle routing. Instances ry48p and kro124p are perturbed random Euclidean instances.

The names of the $1|s_{vu}|C_{\max}$ problems, their dimensions and optimal values $C^*_{\max}$ of makespan function are listed in Tables 1, 2 and 3. The optimal solutions to ATSP instances may be found in [11]. To find the optimal solutions to $1|s_{vu}|C_{\max}$ instances, we employed CPLEX MIP solver with addition of problem specific cuts which were constructed using the well-known approach [24].

Table 1: Instances of $1|s_{vu}|C_{\max}$ problems in series ftv

instance	ftv33	ftv35	ftv38	ftv44	ftv47	ftv55	ftv64	ftv70	ftv90
k	34	36	39	45	48	56	65	71	91
$C^*_{\max}$	1159	1323	1399	1488	1634	1485	1656	1818	1482
instance	ftv100	ftv110	ftv120	ftv130	ftv140	ftv150	ftv160	ftv170	
k	101	111	121	131	141	151	161	171	
$C^*_{\max}$	1691	1857	2023	2189	2320	2511	2561	2642	

Table 2: Instances of $1|s_{vu}|C_{\max}$ problems in series rbg

instance	rbg323	rbg358	rbg403	rbg443
k	323	358	403	443
$C^*_{\max}$	1299	1130	2432	2687

The experiment consisted of two stages. On the first stage, the competing GAs were run for a given number of iterations in order to estimate the influence of different crossover

Table 3: Other $1|s_{vu}|C_{\max}$ instances

instance	ry48p	ft53	ft70	kro124p
k	48	53	70	100
$C^*_{\max}$	13451	5846	36981	35227

operators upon the CPU cost of one GA iteration. Besides that, the shortest average execution time, denoted $t_{\min}$, was identified for each problem instance. On the second stage of experiments, a number of independent runs of competing GAs were made with each instance, given the time budget $t_{\min}$ seconds for each run. This stage was aimed at evaluation of frequency of finding optimal solutions.

4.3 Makespan Minimization Problem on a Single Machine

First we consider $1|s_{vu}|C_{\max}$ problem. In what follows, GA1 denotes the GA with position-based representation employing the Optimized Cycle Crossover. We use the notation GA2 for the GA with adjacency-based representation where the ORP is solved approximately with only one application of the algorithm of D. Eppstein. This corresponds to testing at most two options for vertex v among the vertices that correspond to the last jobs of parent schedules. Exact solving of this ORP requires enumeration of $O(k)$ options for vertex v and it was not used in the experiments due to high computational burden.

On the first stage of the experiment, the GA with elitist recombination was run 1000 times for each instance and each run continued for 4000 iterations for all problems except rbg series. In the case of rbg series, each run continued for 8000 iterations. These termination conditions were chosen on the basis of preliminary experiments which showed that such numbers of iterations were enough for GA1 to find the optimal solutions with a sufficiently high probability (more than 5%).

Average execution times of GAs (in seconds) denoted by $t^{\text{avr}}_{\text{GA1}}$ and $t^{\text{avr}}_{\text{GA2}}$ are shown in Tables 4 and 5.

Table 4: Average execution time for $1|s_{vu}|C_{\max}$ instances in series ftv

instance	ftv33	ftv35	ftv38	ftv44	ftv47	ftv55	ftv64	ftv70	ftv90
$t^{\text{avr}}_{\text{GA1}}$	**0.26**	**0.25**	**0.27**	**0.29**	**0.3**	0.75	**0.72**	**0.95**	**1.27**
$t^{\text{avr}}_{\text{GA2}}$	0.35	0.38	0.44	0.51	0.58	**0.71**	0.87	1.15	1.49
instance	ftv100	ftv110	ftv120	ftv130	ftv140	ftv150	ftv160	ftv170	
$t^{\text{avr}}_{\text{GA1}}$	**1.59**	**1.93**	2.93	4.34	5.07	**3.76**	**4.41**	**4.52**	
$t^{\text{avr}}_{\text{GA2}}$	1.8	2.5	**2.84**	**3.23**	**4.09**	4.88	5.82	8.51	

Table 5: Average execution time for other $1|s_{vu}|C_{\max}$ instances

instance	ry48p	ft53	ft70	kro124p
$t^{\text{avr}}_{\text{GA1}}$	**0.43**	**0.32**	**0.35**	8.1
$t^{\text{avr}}_{\text{GA2}}$	0.66	0.67	1.03	**2.03**

On majority of the problem instances (16 out of 21) presented in Tables 4 and 5, algorithm GA1 terminated faster compared to GA2. However the average execution time of GA2 is at most twice the average execution time of GA1 on all instances, except ft53 and ft70.

On the second stage of experiments, both GAs were run 1000 times for equal amount of time $t_{\min} = \min\{t^{\text{avr}}_{\text{GA1}}, t^{\text{avr}}_{\text{GA2}}\}$. The results of this stage are displayed in Tables 6 and 7. Here $F^{\text{opt}}_{\text{GA1}}$ and $F^{\text{opt}}_{\text{GA2}}$ are the frequencies of finding an optimum for GA1 and GA2 (respectively).

The statistical analysis of experimental data was carried out using the following approach. For each problem, the testing of an algorithm is considered as a sequence of ν Bernoulli trials, where "success" corresponds to finding an optimal solution. In our experiments, we performed $\nu = 1000$ trials with GA1 and GA2 algorithms. The confidence intervals for the success probability p^* are built using the standard method [25] applied to the Bernoulli distribution and presented in Tables 6 and 7 (the confidence level is set to 5%). These tables show that on most of the instances the considered GAs have similar performance.

On the first stage of experiments with rbg series the execution time of GA2 was much longer than that of GA1. Therefore both algorithms were given $t^{\text{avr}}_{\text{GA1}}$ seconds on the second stage. The execution times $t^{\text{avr}}_{\text{GA1}}$ and the frequencies of finding an optimum in 1000 runs are presented in Table 8. GA1 has a significant advantage on this series, which is due to large computational cost of crossover in GA2.

We carried out additional experiment in order to find out whether this drastic difference in computational cost of the two crossovers is due to specific problems structure in rbg series or it is due to high dimension of these problems. To this end, we generated testing instances with the same numbers of jobs as in rbg and chose the setup times uniformly from $[s_{\max}/2, s_{\max}]$, where $s_{\max}$ is the largest setup time on the corresponding instance from series rbg. It turned out that there was no drastic difference between the CPU times of GA1 and GA2 in this additional experiment, so we conclude that the cause of poor performance of GA1 on series rbg might be in the specific structure of these instances.

As seen from the tables, GA1 in general demonstrates more stable results than GA2.

The dynamics of CPU time required for solving ORPs in GA1 and GA2 on instances ftv100, ftv130, ftv150, ftv170 is displayed in Fig. 1. The plots for the remaining problems of ftv series were analogous and they are skipped here. The CPU time required for solving ORPs in GA1 on series rbg is shown in Fig. 2. It can be seen that the time-complexity of

Table 6: Frequencies of finding the optimum and confidence intervals for $1|s_{vu}|C_{\max}$ instances in series ftv

instance	ftv33	ftv35	ftv38	ftv44	ftv47	ftv55
$F^{\text{opt}}_{\text{GA1}}$	**0.69**	**0.6**	**0.6**	**0.6**	**0.5**	0.49
$I^{\text{conf}}_{\text{GA1}}$	(0.66;0.72)	(0.57;0.63)	(0.57;0.63)	(0.57;0.63)	(0.47;0.53)	(0.46;0.52)
$F^{\text{opt}}_{\text{GA2}}$	0.65	0.59	0.59	0.44	0.4	**0.5**
$I^{\text{conf}}_{\text{GA2}}$	(0.62;0.68)	(0.56;0.62)	(0.56;0.62)	(0.41;0.47)	(0.37;0.43)	(0.47;0.53)
instance	ftv64	ftv70	ftv90	ftv100	ftv110	ftv120
$F^{\text{opt}}_{\text{GA1}}$	**0.49**	0.51	**0.4**	**0.37**	0.31	0.24
$I^{\text{conf}}_{\text{GA1}}$	(0.46;0.52)	(0.48;0.54)	(0.37;0.43)	(0.34;0.4)	(0.28;0.34)	(0.21;0.27)
$F^{\text{opt}}_{\text{GA2}}$	**0.49**	**0.53**	0.39	0.33	**0.35**	**0.27**
$I^{\text{conf}}_{\text{GA2}}$	(0.46;0.52)	(0.5;0.56)	(0.36;0.42)	(0.3;0.36)	(0.32;0.38)	(0.24;0.3)
instance	ftv130	ftv140	ftv150	ftv160	ftv170	
$F^{\text{opt}}_{\text{GA1}}$	0.27	0.31	0.29	**0.4**	**0.36**	
$I^{\text{conf}}_{\text{GA1}}$	(0.24;0.3)	(0.28;0.34)	(0.26;0.32)	(0.37;0.43)	(0.33;0.39)	
$F^{\text{opt}}_{\text{GA2}}$	**0.31**	**0.41**	**0.31**	0.39	0.3	
$I^{\text{conf}}_{\text{GA2}}$	(0.28;0.34)	(0.38;0.44)	(0.28; 0.34)	(0.36;0.42)	(0.27;0.33)	

Table 7: Frequencies of finding the optimum and confidence intervals for other $1|s_{vu}|C_{\max}$ instances

instance	ry48p	ft53	ft70	kro124p
$F^{\text{opt}}_{\text{GA1}}$	**0.4**	**0.55**	**0.43**	0.22
$I^{\text{conf}}_{\text{GA1}}$	(0.37;0.43)	(0,52;0,58)	(0.4;0.46)	(0.19;0.25)
$F^{\text{opt}}_{\text{GA2}}$	**0.4**	0.35	0.32	**0.53**
$I^{\text{conf}}_{\text{GA2}}$	(0.37;0.43)	(0.32;0.38)	(0.29;0.35)	(0.5;0.56)

Table 8: Average time and frequency of finding the optimum for $1|s_{vu}|C_{\max}$ instances in series rbg

instance	rbg323	rbg358	rbg403	rbg443
$t^{\text{avr}}_{\text{GA1}}$	7.02	7.47	7.54	7.53
$F^{\text{opt}}_{\text{GA1}}$	**0.195**	**0.111**	**0.107**	**0.091**
$F^{\text{opt}}_{\text{GA2}}$	0	0	0	0

crossover operators decreases with iterations count, which is due to decreasing population diversity. The CPU cost of optimized crossover in the case of the position-based representation is somewhat smaller compared to the adjacency-based representation. The ORPs

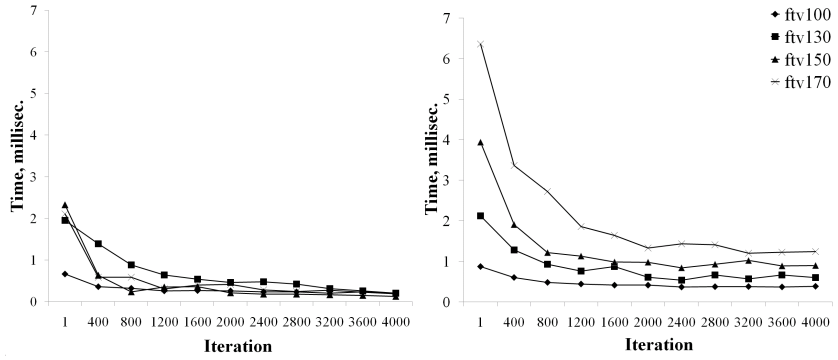

Figure 1: Average CPU time of crossover on $1|s_{vu}|C_{\max}$ instances in series ftv. The left plot corresponds to GA1, the right plot corresponds to GA2.

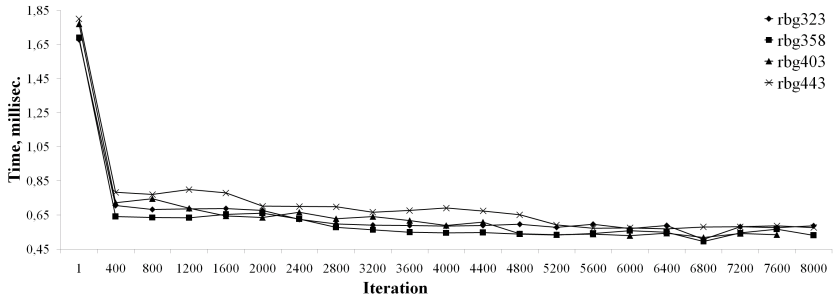

Figure 2: Average CPU time of crossover in GA1 on series rbg.

for instance ftv170 are especially hard for GA2. This observation agrees with the greater execution time of GA2 and its lower frequency of obtaining the optimum on ftv170.

It was mentioned in Subsection 2.1 that with probability approaching to 1, as $k \to \infty$, randomly chosen parent solutions define an ORP instance with "good" graph G and the Optimized Cycle Crossover requires $O(k\ln(k))$ time. The high frequency of such ORP instances was observed in the experiments, e.g. on ftv series the "good" graphs G were observed in more than 60% of crossover calls, and in more than 80% of the calls on rbg series. In process of GA execution this frequency increased.

4.4 Asymmetric Travelling Salesman Problem

Experiments with ATSP were carried out following the same outline as with $1|s_{vu}|C_{\max}$ on instances of TSPLIB. In what follows, GA1' denotes the GA for ATSP based on Optimized Cycle Crossover and GA2' denotes the GA for ATSP based on Optimized Directed Edge Crossover, where the ORP with adjacency-based representation is solved

exactly (see Section 3).

Average execution times of GA1' and GA2' ($t^{\text{avr}}_{\text{GA1'}}$ and $t^{\text{avr}}_{\text{GA2'}}$) are close to those of GA1 and GA2 respectively. A rough comparison on the basis of computers performance table [26] suggests that the CPU resource given to GAs in our experiments is approximately 3 times the resource used by SAX/RAI memetic algorithm in [8] on all instances, except for series rbg. The latter series is excluded in this comparison because in [8], a problem-specific heuristic of W. Zhang [27] was used in construction of initial populations. This heuristic of W. Zhang is very efficient on series rbg and most likely the optimal solutions to all rbg instances were found in SAX/RAI memetic algorithm at the initialization stage.

Tables 9, 10 and 11 present the frequencies of finding an optimum in 1000 runs, given $t_{\min} = \min\{t^{\text{avr}}_{\text{GA1'}}, t^{\text{avr}}_{\text{GA2'}}\}$ CPU seconds for each run, and the confidence intervals for the probability of obtaining an optimum (the confidence level is 5%).

Table 9: Frequencies of finding the optimum and confidence intervals for ATSP series ftv

instance	ftv33	ftv35	ftv38	ftv44	ftv47	ftv55
$F^{\text{opt}}_{\text{GA1'}}$	0.51	0.53	0.52	0.51	0.47	0.4
$I^{\text{conf}}_{\text{GA1'}}$	(0.48;0.54)	(0.5;0.56)	(0.49;0.55)	(0.48;0.54)	(0.44;0.5)	(0.37;0.43)
$F^{\text{opt}}_{\text{GA2'}}$	**0.93**	**0.76**	**0.75**	**0.7**	**0.86**	**0.67**
$I^{\text{conf}}_{\text{GA2'}}$	(0.91;0.95)	(0.73;0.79)	(0.72;0.78)	(0.67;0.73)	(0.84;0.88)	(0.64;0.7)
instance	ftv64	ftv70	ftv90	ftv100	ftv110	ftv120
$F^{\text{opt}}_{\text{GA1'}}$	0.4	0.39	0.35	0.33	0.29	0.22
$I^{\text{conf}}_{\text{GA1'}}$	(0.37;0.43)	(0.36;0.42)	(0.32;0.38)	(0.3;0.36)	(0.26;0.32)	(0.19;0.25)
$F^{\text{opt}}_{\text{GA2'}}$	**0.79**	**0.65**	**0.38**	**0.47**	**0.32**	**0.26**
$I^{\text{conf}}_{\text{GA2'}}$	(0.76;0.82)	(0.62;0.68)	(0.35;0.41)	(0.44;0.5)	(0.29;0.35)	(0.23;0.29)
instance	ftv130	ftv140	ftv150	ftv160	ftv170	
$F^{\text{opt}}_{\text{GA1'}}$	0.29	0.2	0.22	0.38	**0.31**	
$I^{\text{conf}}_{\text{GA1'}}$	(0.26;0.32)	(0.17;0.23)	(0.19;0.25)	(0.35;0.41)	(0.28;0.34)	
$F^{\text{opt}}_{\text{GA2'}}$	**0.41**	**0.42**	**0.43**	**0.41**	0.3	
$I^{\text{conf}}_{\text{GA2'}}$	(0.38;0.44)	(0.39;0.45)	(0.4;0.46)	(0.38;0.44)	(0.27;0.33)	

Table 10: Frequencies of finding the optimum and confidence intervals for other ATSP instances

instance	ry48p	ft53	ft70	kro124p
$F^{\text{opt}}_{\text{GA1'}}$	0.37	0.53	**0.42**	0.1
$I^{\text{conf}}_{\text{GA1'}}$	(0.34;0.4)	(0.5;0.56)	(0.39;0.45)	(0.08;0.12)
$F^{\text{opt}}_{\text{GA2'}}$	**0.42**	**0.64**	**0.42**	**0.47**
$I^{\text{conf}}_{\text{GA2'}}$	(0.39;0.45)	(0.61;0.67)	(0.39;0.45)	(0.44;0.5)

Table 11: Frequency of finding the optimum for ATSP series rbg

instance	rbg323	rbg358	rbg403	rbg443
$F^{\text{opt}}_{\text{GA1'}}$	**0.145**	**0.105**	**0.086**	**0.079**
$F^{\text{opt}}_{\text{GA2'}}$	0	0.001	0	0

On majority of the problems (19 out of 25) GA2' finds an optimum more frequently than GA1' (in 14 cases among these the confidence intervals for p^* do not intersect), although GA1' is still more successful on rbg series. Better results of GA2' with adjacency-based representation in the case of ATSP, compared to the results of GA2 on $1|s_{vu}|C_{\max}$ problem, are presumably due to exact solving of the ORP in Optimized Directed Edge Crossover.

Comparing GA2' and SAX/RAI memetic algorithm from [8] in terms of frequencies of finding optimal solutions, we estimate the frequency of GA2' as approximately 70% of the frequency reported for SAX/RAI memetic algorithm on all instances, except for series rbg. This outcome seems to be promising since the general GA outline and tunable parameters were chosen quite straightforwardly in this paper.

Summing up the experimental results for $1|s_{vu}|C_{\max}$ problem and ATSP in terms of frequency of finding optimal solutions we can conclude that the two compared approaches are competitive with each other. In the case of $1|s_{vu}|C_{\max}$ problem, GA1 tends to outperform GA2 on larger instances such as ftv170, rbg323, rbg358, rbg403 and rbg443. In the case of ATSP, GA2' dominates GA1', except for series rbg where instances have special structure.

We carried out an additional experiment in order to compare the optimized crossovers ODEC and OCX to their randomized prototypes DEC and RCX. It clearly showed an advantage of ODEC and OCX over DEC and RCX. For the large-scale problems such as ftv110, ftv120, ftv150, kro124p, rbg323, and rbg358 the GA with operators DEC and RCX found optimal solution within the same CPU time limit $t_{\min}$ no more than once out of 1000 runs. A similar situation was observed in the case of $1|s_{vu}|C_{\max}$ problem.

5 Conclusions

Optimal recombination problems for Makespan Minimization Problem on a Single Machine and for Asymmetric Travelling Salesman Problem are shown to be NP-hard under two "natural" solutions encodings (position-based representation and adjacency-based representation). In the case of position-based representation, almost all instances of the optimal recombination problem are polynomially solvable both for $1|s_{vu}|C_{\max}$ and ATSP. The worst case time-complexity of optimized crossover operators is $O(k^2 2^{\frac{k}{2}})$ in the case of adjacency-based representation for $1|s_{vu}|C_{\max}$ and it is $O(k2^{\frac{k}{2}})$ (or $O(n2^{\frac{n}{2}})$)

in the other cases considered in this paper. The computational experiment indicates that the two approaches to optimal recombination yield competitive results. However GA with position-based representation dominates GA with adjacency-based representation on problems with special structure.

Further research might extend the analysis to other problems on permutations and reduce some of the known upper bounds on the time complexity of optimal recombination. In particular, we hypothesize that the time complexity of the optimized crossover for $1|s_{vu}|C_{\max}$ with adjacency-based representation may be reduced to $O(k2^{\frac{k}{2}})$. We expect that the GA behavior observed in this paper might be helpful for improvement of state-of-the-art metaheuristics for problems on permutations.

Acknowledgements

This research is supported by the Russian Science Foundation grant 15-11-10009.

References

[1] Radcliffe, N.J.: The Algebra of Genetic Algorithms. Annals of Mathemathics and Artificial Intelligence **10** (4), 339–384 (1994)

[2] Yagiura, M., Ibaraki, T.: The Use of Dynamic Programming in Genetic Algorithms for Permutation Problems. Eur. Jour. Oper. Res. **92**, 387–401 (1996)

[3] Cotta, C., Alba, E., Troya, J. M.: Utilizing Dynastically Optimal Forma Recombination in Hybrid Genetic Algorithms. Proc. of 5-th Int. Conf. on Parallel Problem Solving from Nature. LNCS, Vol. 1498, 305–314, Springer, Berlin (1998)

[4] Cook, W., Seymour, P.: Tour Merging via Branch-Decomposition. INFORMS Journal on Computing **15** (2), 233–248 (2003)

[5] Tinós, R., Whitley, D., Ochoa, G.: Generalized Asymmetric Partition Crossover (GAPX) for the Asymmetric TSP. Proc. of the 2014 Annual Conference on Genetic and Evolutionary Computation, 501–508, ACM New York, New York (2014)

[6] Cotta, C., Troya, J.M.: Genetic Forma Recombination in Permutation Flowshop Problems. Evolutionary Computation **6** (1), 25–44 (1998)

[7] Whitley, D., Starkweather, T., Shaner, D.: The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination. In L. Davis (ed.), Handbook of Genetic Algorithms, 350–372, Van Nostrand Reinhold, New York (1991)

[8] Buriol, L.S., Franca, P.M., Moscato, P.: A New Memetic Algorithm for the Asymmetric Traveling Salesman Problem. Journal of Heuristics **10**, 483–506 (2004)

[9] Cirasella, J., Johnson, D.S., McGeoch, L.A., Zhang, W.: The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators, and Tests. In A.L. Buchsbaum and J. Snoeyink, (eds.), Proc. of ALENEX01. LNCS, Vol. 2153, 32–59, Springer, Berlin(2001)

[10] Nagata, Y., Soler, D.: A New Genetic Algorithm for the Asymmetric Travelling Salesman Problem. Expert Syst. with Applications **39** (10), 8947-–8953 (2012)

[11] Reinelt, G.: TSPLIB – A Traveling Salesman Problem Library. ORSA Journal on Computing **3** (4), 376–384 (1991)

[12] Holland, J.: Adaptation in Natural and Artificial Systems. Ann Arbor, University of Michigan Press (1975)

[13] Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NP-completeness. W.H. Freeman and Company, San Francisco (1979)

[14] Eremeev, A.V., Kovalenko, J.V.: Optimal Recombination in Genetic Algorithms for Combinatorial Optimization Problems: Part II. Yugoslav Journal of Operations Research **24** (2), 165–186 (2014)

[15] Serdyukov, A.I.: On Travelling Salesman Problem with Prohibitions. Upravlaemye systemi **1**, 80–86 (1978) (In Russian)

[16] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. 2nd edition, MIT Press (2001)

[17] Hazir, Ö., Günalay, Y., Erel, E.: Customer Order Scheduling Problem: A Comparative Metaheuristics Study. Int. Journ. of Adv. Manuf. Technol. **37**, 589–598 (2008)

[18] Chvatal, V.: Probabilistic Methods in Graph Theory. Annals of Operations Research **1**, 171–182 (1984)

[19] Eppstein, D.: The Traveling Salesman Problem for Cubic Graphs. Journal of Graph Algorithms and Applications **11** (1), (2007)

[20] Goldberg, D., Thierens, D.: Elitist Recombination: An Integrated Selection Recombination GA. Proc. first IEEE World Congress on Computational Intelligence. Vol. 1, 508–512, Piscataway, New Jersey: IEEE Service Center (1994)

[21] Fischetti, M., Toth, P.: An Additive Bounding Procedure for the Asymmetric Travelling Salesman Problem. Math. Programming A **53**, 173–197 (1992)

[22] Fischetti, M., Toth, P.: A Polyhedral Approach to the Asymmetric Travelling Salesman Problem. Management Sci. **43**, 1520–1536 (1997)

[23] Fischetti, M., Toth, P., Vigo, D.: A Branch and Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs. Oper. Res. **42**, 846–859 (1994)

[24] Dantzig, G., Fulkerson, R., Johnson, S.: Solution of a Large-Scale Traveling Salesman Problem. Oper. Res. **2**, 393–410 (1954)

[25] Mood, A.M., Graybill, F.A., Boes, D.C.: Introduction to the Theory of Statistics. 3d edition, New York, McGraw-Hill (1973)

[26] Dongarra, J.J.: Performance of Various Computers Using Standard Linear Equations Software. Technical Report No. CS-89-85, University of Manchester, 110 p. (2014)

[27] Zhang, W.: Depth-First Branch-and-Bound versus Local Search: A Case Study. Proc. of 17th National Conf. on Artificial Intelligence 930-–935, Austin, TX (2000)