Yugoslav Journal of Operations Research
24 (2014) Number 2,165-186
DOI: 10.2298/YJOR131030041E

OPTIMAL RECOMBINATION IN GENETIC ALGORITHMS FOR COMBINATORIAL OPTIMIYATION PROBLEMS – PART II

Anton V. EREMEEV
Sobolev Institute of Mathematics, Laboratory of Discrete Optimization
630090, Novosibirsk, Russia
eremeev@ofim.oscsbras.ru
Julia V. KOVALENKO
Omsk F.M. Dostoevsky State University,
Institute of Mathematics and Information Technologies,
644077, Omsk, Russia
juliakoval86@mail.ru

Received: July 2013 / Accepted: October 2013

Abstract: This paper surveys results on complexity of the optimal recombination problem (ORP), which consists in finding the best possible offspring as a result of a recombination operator in a genetic algorithm, given two parent solutions. In Part II, we consider the computational complexity of ORPs arising in genetic algorithms for problems on permutations: the Travelling Salesman Problem, the Shortest Hamilton Path Problem and the Makespan Minimization on Single Machine and some other related problems. The analysis indicates that the corresponding ORPs are NP-hard, but solvable by faster algorithms, compared to the problems they are derived from.
Keywords: Genetic Algorithm, Optimal Recombination Problem, complexity, crossover, permutation problems.
MSC: 90C59, 90C10.

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. This survey is devoted to complexity and solution methods of the *Optimal Recombination Problem* (ORP), which consists in finding the best possible offspring as a result of a crossover operator, given two feasible parent solutions (see Part I for a formal definition). The experimental results of M. Yagiura and T. Ibaraki [30], C. Cotta, E. Alba and J.M. Troya [8], W. Cook and P. Seymour [6], D. Whitley, D. Hains and A. Howe [29] indicate that optimal recombination may be used successfully in the GAs for problems on permutations.

Part II of this survey is structured as follows. The computational complexity of ORP for the Travelling Salesman Problem is considered in Section 1 both for the symmetric case and for the general case. Strong NP-hardness of these optimal recombination problems is proven and solving approaches are proposed. The ORP for Shortest Hamiltonian Path Problem is shown to be strongly NP-hard as well. A closely related problem of Makespan Minimization on Single Machine is considered in Section 2: it is shown that on one hand the ORP for this problem is strongly NP-hard, on the other hand, almost all instances of this ORP are efficiently solvable. Sections 3 and 4 are devoted to issues for further research and concluding remarks.

1 TRAVELLING SALESMAN PROBLEM

In this section we consider the Travelling Salesman Problem (TSP): suppose a digraph G without loops or multiple arcs is given. The set of vertices of G is V and a set of arcs is A. A weight (length) $c_{ij} \geq 0$ of each arc $(i, j) \in A$ is given as well. It is required to find a Hamiltonian circuit of minimum length.

If for each arc $(i, j) \in A$ there exists a reverse one $(j, i) \in A$ and $c_{ij} = c_{ji}$, then the TSP is called *symmetric* and G is assumed to be an ordinary graph. Without this assumption, we will call the problem the *general case* of TSP.

Feasible solution to the TSP may be encoded as a sequence of the vertex numbers in the TSP tour, or as a permutation matrix where the element in row i and column j equals one iff the vertex j immediately follows the vertex i in the TSP tour. (For the sake of consistency with definition of NP optimization problem (see Part I or [2]), one may assume that the elements of the matrix are written sequentially in a string $\mathbf{x} \in \text{Sol}$.)

Unfortunately, there are $|V|$ different sequences of vertices encoding the same Hamiltonian circuit. The second encoding has an advantage that a Hamiltonian circuit is uniquely represented by a permutation matrix. Therefore, in what follows we assume the second encoding. If this encoding is used in the symmetric case, it is sufficient to define only the elements above the diagonal of the matrix, so the rest of the elements are dismissed from subsequent consideration in the symmetric case.

The encoding by permutation matrix defines an ORP that consists in finding a shortest travelling salesman's tour which coincides with two given feasible parent solutions in those arcs (or edges), which belong to both parent tours and does not contain the arcs (or edges) which are absent in both parent solutions.

1.1 Symmetric Case

In [21] it is proven that recognition of Hamiltonian grid graphs (the *Hamilton Cycle Problem*) is NP-complete. Recall that a graph $G' = (V', E')$ with vertex set V' and edge set E' is called a *grid* graph, if its vertices are the integer vectors $v = (x_v, y_v) \in \mathbf{Z}^2$ on plane, i.e., $V' \subset \mathbf{Z}^2$, and a pair of vertices is connected by an edge iff the Euclidean distance between them is equal to 1. Here and below, $\mathbf{Z}$ denotes the set of integer numbers. Let us call the edges that connect two vertices in $\mathbf{Z}^2$ with equal first coordinates *vertical edges.* The edges that connect two vertices in $\mathbf{Z}^2$ with equal second coordinates will be called *horizontal edges.*

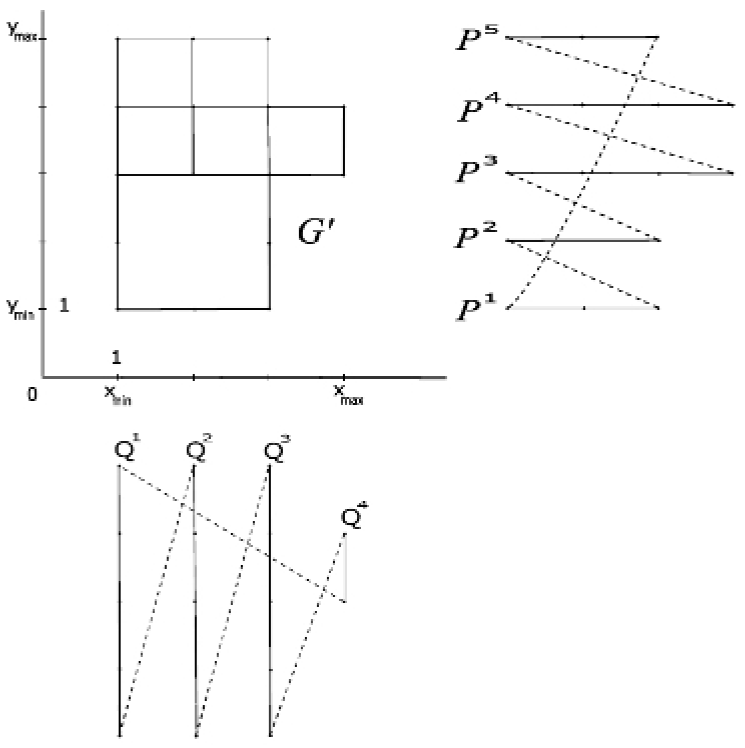

Figure 1: Example of two parent tours used in reduction from Hamilton Cycle Problem to ORP in symmetric case.

Let us assume that $V' > 4$, graph G' is connected and there are no bridges in G' (note that if any of these assumptions is violated, then existence of a Hamiltonian cycle in G' can be recognized in polynomial time). Now, we will construct a reduction from the Hamilton Cycle Problem for G' to an ORP for a complete edge-weighted graph $G = (V, E)$, where $V = V'$.

Let the edge weights c_{ij} in G be defined so that if a pair of vertices $\{v_i, v_j\}$ is connected by an edge of G', then $c_{ij} = 0$; all other edges in G have the weight 1. Consider the following two parent solutions of the TSP on graph G (an example of graph G' and two parent solutions for the corresponding TSP is given in Fig. 1).

Let $y_{\min} = \min_{v \in V'} y_v$, $y_{\max} = \max_{v \in V'} y_v$. For any integer $y \in \{y_{\min}, \dots, y_{\max}\}$, the horizontal chain that passes through vertices $v \in V'$ with $y_v = y$ by increasing values of coordinate x is denoted by P^y. Let the first parent tour follow the chains $P^{y_{\min}}, P^{y_{\min}+1}, \dots, P^{y_{\max}}$, connecting the right-hand end of each chain P^y with $y < y_{\max}$ to the left-hand end of the chain P^{y+1}. Note that these connections never coincide with vertical edges because G' has no bridges. To create a cycle, connect the right-hand end v_{TR} of the chain $P^{y_{\max}}$ to the left-hand end v_{BL} of the chain $P^{y_{\min}}$.

The second parent tour is constructed similarly using the vertical chains. Let $x_{\min} = \min_{v \in V'} x_v$, $x_{\max} = \max_{v \in V'} x_v$. For any integer $x \in \{x_{\min}, \dots, x_{\max}\}$, the vertical chain that passes monotonically in y through the vertices $v \in V'$, such that $x_v = x$, is denoted by Q^x. The second parent tour follows the chains $Q^{x_{\min}}, Q^{x_{\min}+1}, \dots, Q^{x_{\max}}$, connecting the lower end of each chain Q^x with $x < x_{\max}$ to the upper end of chain Q^{x+1}. These connections never coincide with horizontal edges since G' has no bridges. Finally, the lower end v_{RB} of chain $Q^{x_{\max}}$ is connected to the upper end v_{LT} of chain $Q^{x_{\min}}$.

Note that the constructed parent tours have no common edges. Indeed, common slanting edges do not exist since $V' > 4$. The horizontal edges belong to the first tour only, except for the situation where $y_{v_{\text{RB}}} = y_{v_{\text{LT}}}$ and the edge $\{v_{\text{RB}}, v_{\text{LT}}\}$ of the second tour is oriented horizontally. But if the first parent tour included the edge $\{v_{\text{RB}}, v_{\text{LT}}\}$ in this situation, then the edge $\{v_{\text{RB}}, v_{\text{LT}}\}$ would be a bridge in graph G'. Therefore, the parent tours can not have the common horizontal edges. Similarly, the vertical edges belong to the second tour only, except for the case where $x_{v_{\text{TR}}} = x_{v_{\text{BL}}}$ and the edge $\{v_{\text{TR}}, v_{\text{BL}}\}$ of the first tour is oriented vertically. But in this case the parent tour can not contain the edge $\{v_{\text{TR}}, v_{\text{BL}}\}$, since G' has no bridges.

Note also that the union of edges of parent solutions contains E'. Consequently, any Hamiltonian cycle in graph G' is a feasible solution of the ORP. At the same time, a feasible solution of the ORP has zero value of objective function iff it contains only the edges of E'. Therefore, the optimal value of objective function in the ORP under consideration is equal to 0 iff there exists a Hamiltonian cycle in graph G'. So, the following theorem is proven.

Theorem 1.1 [13] *Optimal recombination for the TSP in the symmetric case is strongly NP-hard.*

In [21] it is also proven that recognition of grid graphs with a *Hamiltonian path* is NP-complete. Optimal recombination for this problem consists in finding a shortest Hamiltonian path, which uses those edges where both parent tours coincide, and does not use the edges absent in both parent tours. The following theorem is proved analogously to Theorem 1.1.

Theorem 1.2 [13] *Optimal recombination for the problem of finding the shortest Hamiltonian path in a graph with arbitrary edge lengths is strongly NP-hard.*

Note that in the proof of Theorem 1.2, unlike in Theorem 1.1, it is impossible simply to exclude the cases where graph G' has bridges. Instead, the reduction should treat separately each maximal (by inclusion) subgraph without bridges.

Many scheduling problems with setup times contain the problem of finding the shortest Hamiltonian path in a digraph as a special case. In this case the vertices correspond to jobs, the arcs correspond to setups and the arc lengths define the setup times. In view of numerous applications of scheduling problems with setup times, in Section 2 the problem of finding the shortest Hamiltonian path in a digraph is treated as a scheduling problem.

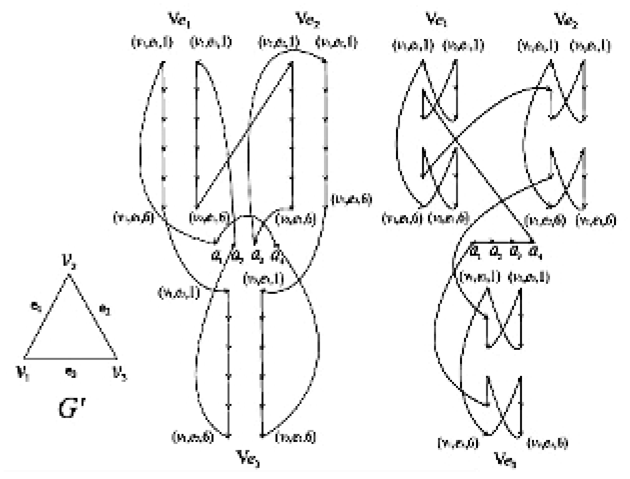

Figure 2: A pair of parent circuits for the case of $G' = K_3$. It is supposed that the incident edges are enumerated as follows. For vertex v^1 : $e^{v_1,1} = e_1$, $e^{v_1,2} = e_3$; for vertex v^2 : $e^{v_2,1} = e_1$, $e^{v_2,2} = e_2$; for vertex v^3 : $e^{v_3,1} = e_2, e^{v_3,2} = e_3$.

1.2 The General Case

In the general case of TSP, the ORP is not a more general problem than the ORP considered in Subsection 1.1 because in the problem input we have two directed parent paths while in the symmetric case, the parent paths were undirected. Even if the distance matrix (c_{ij}) is symmetric, a pair of directed parent tours defines a significantly different set of feasible solutions, compared to the undirected case. Therefore, the general case requires a separate consideration of ORP complexity.

Theorem 1.3 [13] *Optimal recombination for the TSP in the general case is strongly NP-hard.*

Proof. We use a modification of the textbook reduction of the Vertex Cover Problem to the TSP [17].

Suppose an instance of a Vertex Cover Problem is given as a graph $G' = (V', E')$. It is required to find a vertex cover of minimal size in G'. Let us assume that the vertices in V' are enumerated, i.e. $V' = \{v_1, \dots, v_n\}$, where $n = |V'|$, and let $m = |E'|$.

Consider a complete digraph $G = (V, A)$ where the set of vertices V consists of $|E'|$ *cover-testing* components, each one containing 12 vertices: $V_e = \{(v_i, e, k), (v_j, e, k) : 1 \leq k \leq 6\}$ for each $e = \{v_i, v_j\} \in E'$, $i < j$. Besides that, V contains n *selector* vertices denoted by $a_1, \dots, a_n$, and a *supplementary* vertex a_{n+1}.

Let the parent tours in graph G be the two circuits defined below (an example of a pair of such circuits for the case of $G' = K_3$ is provided in Fig. 2).

1. Each cover-testing component V_e, where $e = \{v_i, v_j\} \in E'$ and $i < j$ is visited twice by the first tour. The first time, it visits the vertices that correspond to v_i in the sequence

$$(v_i, e, 1), \dots, (v_i, e, 6), \tag{1}$$

the second time, it visits the vertices corresponding to v_j, in the sequence

$$(v_j, e, 1), \dots, (v_j, e, 6). \tag{2}$$

2. The second tour goes through each cover-testing component V_e, where $e = \{v_i, v_j\} \in E'$ and $i < j$ in the following sequence:

$$(v_i, e, 2), (v_i, e, 3), (v_j, e, 1), (v_j, e, 2), (v_j, e, 3), (v_i, e, 1),$$
$$(v_i, e, 6), (v_j, e, 4), (v_j, e, 5), (v_j, e, 6), (v_i, e, 4), (v_i, e, 5).$$

The first parent tour connects the cover-testing components as follows. For each vertex $v \in V'$ order arbitrarily the edges incident to v in graph G' in sequence: $e^{v,1}, e^{v,2}, \dots, e^{v,deg(v)}$, where $deg(v)$ is the degree of vertex v in G'. In the cover-testing components, following the chosen sequence $e^{v,1}, e^{v,2}, \dots, e^{v,deg(v)}$, this tour passes 6 vertices in each of the components (v, e, k), $k = 1, \dots, 6$, $e \in \{e^{v,1}, e^{v,2}, \dots, e^{v,deg(v)}\}$. Thus, each vertex of any cover-testing component V_e, $e = \{u, v\} \in E'$ will be visited by one of the two 6-vertex sub-tours.

The second tour passes the cover-testing components in an arbitrary order of edges $V_{e_1}, \dots, V_{e_m}$, entering each component V_{e_k} for any $e_k = \{v_{i_k}, v_{j_k}\} \in E'$, $i_k < j_k$, $k = 1, \dots, m$, via vertex $(v_{i_k}, e_k, 2)$ and exiting through vertex $(v_{i_k}, e_k, 5)$. Thus, a sequence of vertex indices $i_1, \dots, i_m$ is induced (repetitions are possible). In what follows, we will need the beginning i_1 and the end i_m of this sequence.

The parent sub-tours described above are connected to form two Hamiltonian circuits in G using the vertices $a_1, \dots, a_{n+1}$. The first circuit is completed using the arcs

$$\Big(a_1, (v_1, e^{v_1,1}, 1)\Big),\ \Big((v_1, e^{v_1,deg(v_1)}, 6), a_2\Big),$$
$$\Big(a_2, (v_2, e^{v_2,1}, 1)\Big),\ \Big((v_2, e^{v_2,deg(v_2)}, 6), a_3\Big),$$
$$\dots,$$
$$\Big(a_n, (v_n, e^{v_n,1}, 1)\Big),\ \Big((v_n, e^{v_n,deg(v_n)}, 6), a_{n+1}\Big), \Big(a_{n+1}, a_1\Big).$$

The second circuit is completed by the arcs

$$\Big(a_1, a_2\Big), \dots, \Big(a_{n-1}, a_n\Big), \Big(a_n, a_{n+1}\Big),$$
$$\Big(a_{n+1}, (v_{i_1}, e_1, 2)\Big),\ \Big((v_{i_m}, e_m, 5), a_1\Big).$$

Assign unit weights to all arcs $\Big(a_i, (v_i, e^{v_i,1}, 1)\Big)$, $i = 1, \dots, n$ in the complete digraph G. Besides that, assign weight $n+1$ to all arcs of the second tour which are connecting the components $V_{e_1}, \dots, V_{e_m}$, the same weights are assigned to the arcs $\Big(a_{n+1}, (v_{i_1}, e_1, 2)\Big)$ and $\Big((v_{i_m}, e_m, 5), a_1\Big)$. All other arcs in G are given weight 0.

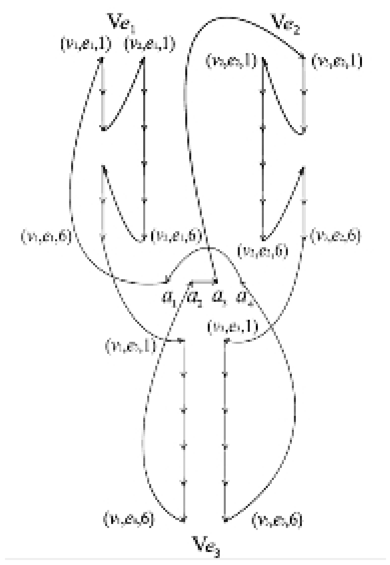

Figure 3: An ORP solution $R(C)$ corresponding to the vertex cover $\{v_1, v_3\}$ of graph $G' = K_3$.

Note that for any vertex cover C of graph G', the set of feasible solutions of ORP with two parents defined above contains a circuit $R(C)$ with the following structure (see an example of such a circuit for the case of $G' = K_3$ in Fig. 3).

For each $v_i \in C$ the circuit $R(C)$ contains the arcs $\Big(a_i, (v_i, e^{v_i,1}, 1)\Big)$ and $\Big((v_i, e^{v_i,deg(v_i)}, 6), a_{i+1}\Big)$. The components V_e, $e \in \{e^{v_i,1}, e^{v_i,2}, \ldots, e^{v_i,deg(v_i)}\}$ are connected together by the arcs from the first tour. For each vertex v_i which does not belong to C, the circuit $R(C)$ has an arc (a_i, a_{i+1}). Also, $R(C)$ passes the arc (a_{n+1}, a_1).

The circuit $R(C)$ visits each cover-testing component V_e by one of the two ways:

1. If both endpoints of an edge e belong to C, then $R(C)$ passes the component following the same arcs as the first parent tour.

2. If $e = \{u, v\}$, $u \in C$, $v \notin C$, then $R(C)$ visits the vertices of the component in sequence

$$(u,e,1),(u,e,2),(u,e,3), \quad (v,e,1),\ldots,(v,e,6), \quad (u,e,4),(u,e,5),(u,e,6).$$

One can check straightforwardly that this sequence does not violate the ORP constraints.

In general, the circuit $R(C)$ is a feasible solution to the ORP because, on one hand, all arcs used in $R(C)$ are present at least in one of the parent tours. On the other hand, both parent tours contain only the arcs of the type

$$\Big((u,e,2),(u,e,3)\Big),\ \Big((u,e,4),(u,e,5)\Big),\ \Big((v,e,1),(v,e,2)\Big),$$
$$\Big((v,e,2),(v,e,3)\Big),\ \Big((v,e,4),(v,e,5)\Big),\Big((v,e,5),(v,e,6)\Big)$$

within the cover-testing components V_e, $e = \{u, v\} \in E'$, where vertex u has smaller index than v. All of these arcs belong to $R(C)$. The total weight of circuit $R(C)$ is $|C|$.

Now, each feasible solution R to the constructed ORP defines a set of vertices $C(R)$ as follows: $v_i,\ i \in \{1,\dots,n\}$ belongs to $C(R)$ iff R contains an arc $\Big(a_i,(v_i,e^{v_i,1},1)\Big)$.

Let us consider only such ORP solutions R that have the objective value at most n. These solutions do not contain the arcs that connect the cover-testing components in the second parent tour. They also do not contain the arcs $\Big(a_{n+1},(v_{i_1},e_1,2)\Big)$ and $\Big((v_{i_m},e_m,5),a_1\Big)$. Note that the set of such ORP solutions is non-empty, e.g. the first parent tour belongs to it.

Consider the case where the arc $\Big(a_i,(v_i,e^{v_i,1},1)\Big)$ belongs to R. Each cover-testing component V_e with $e = \{v_i, v_j\} \in E'$ in this case may be visited in one of the two possible ways: either the same way as in the first parent tour (in this case, v_j must also be chosen into $C(R)$ since R is Hamiltonian), or in the sequence

$$(v_i,e,1),(v_i,e,2),(v_i,e,3),\quad (v_j,e,1),\dots,(v_j,e,6),\quad (v_i,e,4),(v_i,e,5),(v_i,e,6)$$

(in this case, v_j will not be chosen into $C(R)$). In view of the assumption that the arc $\Big(a_i,(v_i,e^{v_i,1},1)\Big)$ belongs to R, the cover-testing components $V_e,\ e \in \{e^{v_i,1}, e^{v_i2},\dots,e^{v_i,deg(v_i)}\}$ are connected by the arcs of the first tour, and besides that, R contains the arc $\Big((v_i,e^{v_i,deg(v_i)},6),a_{i+1}\Big)$. Note that the total length of the arcs in R equals $|C(R)|$, and the set $C(R)$ is a vertex cover in graph G', because the tour R passes each component V_e in a way that guarantees coverage of each edge $e \in E'$.

To sum up, there exists a bijection between the set of vertex covers in graph G' and the set of feasible solutions to the ORP of length at most n. The values of objective functions are not changed under this bijection, therefore the statement of the theorem follows. □

1.3 Transformation of the ORP into TSP on Graphs With Bounded Vertex Degree

In this Subsection, the ORP problems are connected to the TSP on graphs (digraphs) with bounded vertex degree, arbitrary positive edge (arc) weights and a given set of *forced* edges (arcs). It is required to find a shortest Hamiltonian cycle (circuit) in the given graph (digraph) that passes all forced edges (arcs).

1.3.1 General Case

Consider the general case of ORP for the TSP, where we are given two parent tours A_1, A_2 in a complete digraph $G = (V, A)$. This ORP problem may be transformed into the problem of finding a shortest Hamiltonian circuit in a supplementary digraph $G' = (V', A')$. The digraph G' is constructed on the basis of G by excluding the set of arcs $A \backslash (A_1 \cup A_2)$ and contracting each path that belongs to both parent tours into a pseudo-arc of the same length and the same direction as those of the path. The lengths of all other arcs that remained in G' are the same as they were in G. A shortest Hamiltonian circuit C' in G' transforms into an optimum of the ORP problem by substitution of each pseudo-arc in C' with the path that corresponds to it.

Note that there are at most two ingoing arcs and at most two outgoing arcs for each vertex in G'. The TSP on such a digraph is equivalent to the TSP on a cubic digraph $G'' = (V'', A'')$, where each vertex $v \in V'$ is substituted by two vertices $\check{v}, \hat{v}$, connected by an *artificial* arc $(\check{v}, \hat{v})$ of zero length. All arcs that entered v, now enter $\check{v}$, and all arcs that left v are now outgoing from $\hat{v}$. Let an arc $e \in A''$ be forced, if it corresponds to a pseudo-arc in G'. Such arcs $e \in A''$ are called pseudo-arcs as well.

A solution to the TSP problem on digraph G'' may be obtained through enumeration of all feasible solutions to a TSP with forced edges on a supplementary graph $\bar{G} = (V'', \bar{E})$. Here, a pair of vertices u, v is connected iff these vertices were connected by an arc (or a pair of arcs) in the digraph G''. An edge $\{u, v\} \in \bar{E}$ is assumed to be forced if (u, v) or (v, u) is a pseudo-arc or an artificial arc in the digraph G''. A set of forced edges in $\bar{G}$ will be denoted by $\bar{F}$. All Hamiltonian cycles in $\bar{G}$ w.r.t. the set of forced edges may be enumerated by means of the algorithm proposed in [12] in time $O(|V''| \cdot 2^{(|\bar{E}|-|\bar{F}|)/4})$. Then, for each Hamiltonian cycle Q from $\bar{G}$ in each of the two directions we can check if it is possible to pass a circuit in G'' through the arcs corresponding to edges of Q, and if possible, compute the length of the circuit. This takes $O(|V''|)$ time for each Hamiltonian cycle. Note that $|\bar{E}| - |\bar{F}| = d \leq |E'| \leq 2n$, where d is the number of arcs which are present in one of the parents only. Consequently, the time complexity of solving the ORP on graph G is $O(n \cdot 2^{d/4})$, which is $O(n \cdot 1.42^n)$.

Implementation of the method described above may benefit in the cases where the parent solutions have many arcs in common.

1.3.2 Symmetric Case

Suppose the symmetric case takes place and two parent Hamiltonian cycles in graph $G = (V, E)$ are defined by two sets of edges E_1 and E_2. Let us construct a reduction of the ORP in this case to a TSP with a set of forced edges on a graph where the vertex degree is at most 4.

Similar to the general case, the ORP reduces to the TSP on a graph $G' = (V', E')$ obtained from G by exclusion of all edges that belong to $E \backslash (E_1 \cup E_2)$ and contraction of all paths that belong to both parent tours. Here, by contraction we mean the following mapping. Let P_{uv} be a path with endpoints in u and v, such that the edges of P_{uv} belong to $E_1 \cap E_2$ and P_{uv} is not contained in any other path with edges from $E_1 \cap E_2$. Assume that contraction of the path P_{uv} maps all of its vertices and edges into one forced edge $\{u, v\}$ of zero length. All other vertices and edges of the graph remain unchanged. Let F' denote the set of forced edges in G', which are introduced when the contraction is applied to all paths wherever possible.

The vertex degrees in G' are at most 4, and $|V'| \leq n$. If an optimum of the TSP on graph G' with the set of forced edges F' is found, then substitution of all forced edges by the corresponding paths yields an optimal solution to the ORP problem. (Note that the objective functions of these two problems differ by the total length of contracted paths.)

The search for an optimum to the TSP on graph G' may be carried out by means of the randomized algorithm proposed in [12] for solving TSP with forced edges on graphs with vertex degree at most 4. Besides the problem input data this algorithm is given a value p, which sets the desired probability of obtaining the optimum. If $p \in [0, 1)$ is a constant which does not depend on the problem input, then the algorithm has time complexity $O((27/4)^{n/3})$, which is $O(1.89^n)$.

When the crossover operator is used in a GA, an additional parameter P_c may be defined to tune the probability of performing recombination. If such a parameter is given, $P_c \in [0, 1)$, then one may assign $p = P_c$. In case $P_c = 1$, the optimal recombination may be performed using a deterministic modification of the algorithm from [12] (corresponding to $p = 1$) which requires greater computation time.

There may be some room for improvement of the algorithms, proposed in [12] for the TSP on graphs with vertex degrees at most 3 or 4 and forced edges, in terms of the running time. Thus, it seems to be important to continue studying this modification of the TSP.

2 MAKESPAN MINIMIZATION ON SINGLE MACHINE

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

The input consists of a set of jobs $V = \{v_1, \dots, v_k\}$ with positive processing times p_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 $C_{\max}$.

Let $\pi = (\pi_1, \dots, \pi_k)$ denote a permutation of the jobs, i. e. π_i is the i-th job on the machine, $i = 1, \dots, k$. 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 $s(\pi^*)$.

We assume that the binary encoding of solutions to this NP optimization problem is given by a permutation matrix, where the element in row i, column u equals 1 iff the i-th executed job is the job u. For the sake of convenience, however, we will continue referring to feasible solutions in terms of permutations where appropriate.

Note that the permutation matrices could be used for encoding the solutions to problem $1|s_{vu}|C_{\max}$ so that a unit element of the matrix reflects a setup between a pair of jobs (similar to the encoding of TSP solutions in Section **??**). Experimental studies of GAs indicate, however, that the solution encodings based on the sequence of jobs (as the one used in this section) yield better results in solving the scheduling problems [25].

2.1 NP-Hardness of Optimal Recombination

In what follows, we will use some remarkable results known for the Shortest Hamiltonian Path Problem with Vertex Requisitions: given a complete digraph $G = (X, U)$, where $X = \{x_1, \dots, x_n\}$ is the set of vertices, $U = \{(x, y) : \; x, y \in X, x \neq y\}$ is the set of arcs with nonnegative weights $\rho(x, y)$, $(x, y) \in U$. Besides that, a family of vertex subsets (requisitions) $X^i \subseteq X, \; i = 1, \dots, n$, is given, such that:
$\mathcal{C}1$: $|X^i| \leqslant 2$ for all $i = 1, \dots, n$;
$\mathcal{C}2$: $1 \leqslant |\{i : \; x \in X^i, \; i = 1, \dots, n\}| \leqslant 2$ for all $x \in X$;
$\mathcal{C}3$: if $x \in X^i$ and $x \in X^j$, where $i \neq j$, then $|X^i| = |X^j| = 2$, and if $x \in X^i$ for a unique i, then $|X^i| = 1$.

Let F denote the set of the bijections from $X_n = \{1, \dots, n\}$ to X that satisfy the condition $f(i) \in X^i, \; i = 1, \dots, n$, for all $f \in F$. The problem asks for a mapping $f^* \in F$, such that $\rho(f^*) = \min\limits_{f \in F} \rho(f)$, where $\rho(f) = \sum\limits_{i=1}^{n-1} \rho(f(i), f(i+1))$ for $f \in F$. In what follows, this problem is denoted by $\mathcal{I}$.

There always exists at least one feasible solution f^1 to Problem $\mathcal{I}$. Indeed, such a solution exists iff there is a perfect matching W in the bipartite graph $\bar{G} = (X_n, X, \bar{U})$ where the subsets of vertices of bipartition X_n, X have equal size and the set of edges is $\bar{U} = \{(i, x) : i \in X_n, \; x \in X^i\}$. Note that if the degree of a vertex $i \in X_n$ in $\bar{G}$ equals d $(1 \leqslant d \leqslant 2)$ then, in view of conditions $\mathcal{C}2$ and $\mathcal{C}3$, the degree of all vertices adjacent to i is also equal to d. Thus for any $Y \subseteq X_n$ holds $|Y| \leqslant |\{x \in X : x \in X^i, \; i \in Y\}|$ and the existence of W follows from the König-Hall Theorem [5]. Besides that, the perfect matching $W = \{(1, x^1), (2, x^2), \dots, (n, x^n)\} \subseteq \bar{U}$ may be found in polynomial time using the König-Hall Algorithm [5]. A feasible solution to problem $\mathcal{I}$ is obtained assuming $f^1(i) = x^i, \; i = 1, \dots, n$.

It is clear that with $|X^i| = 1, \; i = 1, \dots, n$, the problem $\mathcal{I}$ is trivial, since the feasible solution is unique. Therefore in what follows we shall assume that there exists such $i \in X_n$ that $|X^i| = 2$. Then there is at least one more feasible solution f^2 to the problem $\mathcal{I}$, where $f^2(i) = X^i \backslash \{f^1(i)\}$ for such i that $|X^i| = 2$, and $f^2(i) = f^1(i)$ otherwise.

Let us now proceed to complexity analysis of the ORP for $1|s_{vu}|C_{\max}$. First of all note that the problem $\mathcal{I}$ reduces to it. Indeed, associate each vertex $x_i \in X$ of digraph G to a job v_i, $i = 1, \dots, n$, let the number of jobs be n and let the setup times s_{v_i, v_j} be equal to $\rho(x_i, x_j)$ for all $v_i, v_j \in V$, $i \neq j$. Assuming $\pi^1 = f^1$ and $\pi^2 = f^2$, we obtain a polynomial-time reduction of problem $\mathcal{I}$ to the ORP under consideration. In view of properties of this reduction, if $\mathcal{I}$ were strongly NP-hard, this would imply that the ORP for $1|s_{vu}|C_{\max}$ is strongly NP-hard as well.

In [27], A.I. Serdyukov showed the strong NP-hardness of the TSP with Vertex Requisitions, which is the TSP with a family of requisitions defined as above, except that conditions $\mathcal{C}2$ and $\mathcal{C}3$ are dismissed, and the goal is to find such a mapping $\tilde{f}^*$, that $\tilde{\rho}(\tilde{f}^*) = \min\limits_{f \in F} \tilde{\rho}(f)$, where $\tilde{\rho}(f) = \sum\limits_{i=1}^{n-1} \rho(f(i), f(i+1)) + \rho(f(n), f(1))$ for any $f \in F$. Let us denote this problem by $\tilde{\mathcal{I}}$. In what follows it will be shown via a Turing reduction from problem $\tilde{\mathcal{I}}$ that problem $\mathcal{I}$ is NP-hard in the strong sense.

Proposition 2.1 [16] *The problem $\mathcal{I}$ is strongly NP-hard.*

Proof. Let us show that given an instance of problem $\tilde{\mathcal{I}}$ with a family of requisitions X^i, $i = 1, \dots, n$, it is possible to construct efficiently an equivalent family of requisitions that will satisfy conditions $\mathcal{C}1 - \mathcal{C}3$ or, alternatively, to prove that the instance has no feasible solutions.

The equivalent family of requisitions is constructed by the following sequence of transformations, where the vertices and requisitions are labelled as *fathomed* or *unfathomed*. Initially all vertices and requisitions are labelled as unfathomed.

1. If there exists a vertex $x \in X$ such that $\{i \in X_n : x \in X^i\} = \emptyset$, then problem $\tilde{\mathcal{I}}$ has no feasible solutions. No further transformations required.

2. Perform the following operations until only the two-element requisitions will remail among the unfathomed ones: find an unfathomed subset $X^i = \{x\}$ (i.e. $|X^i| = 1$) and delete the vertex x from the other requisitions it belongs to; in case the resulting family of requisitions contains such X^j that $|X^j| = 0$, this implies that $\tilde{\mathcal{I}}$ has no feasible solutions and no further transformations are required; otherwise, label the vertex x and the subset X^i as fathomed.

3. Perform the following operations until among the unfathomed vertices there will be only the vertices that belong to exactly 2 requisitions and each of these requisitions is of cardinality 2: find an unfathomed vertex x that belongs only to one subset $X^i = \{x, y\}$; if the vertex y also belongs only to the subset X^i, then the instance of $\tilde{\mathcal{I}}$ has no feasible solutions and no further transformations are required; otherwise assume $X^i = \{x\}$ and label the vertex x and the subset X^i as fathomed.

It is clear that the obtained family of requisitions is equivalent to the original one and satisfies conditions $\mathcal{C}1 - \mathcal{C}3$. In sequel, without loss of generality we assume that the family of requisitions in $\tilde{\mathcal{I}}$ satisfies $\mathcal{C}1 - \mathcal{C}3$.

Now let us construct a Turing reduction of problem $\tilde{\mathcal{I}}$ to problem $\mathcal{I}$. Suppose there exists a subroutine $\mathcal{S}$ for solving problem $\mathcal{I}$ with a family of requisitions $\bar{X}^i$, $i = 1, \dots, n$. Let us describe an algorithm $\mathcal{A}$ for solving problem $\tilde{\mathcal{I}}$ with a family of requisitions X^i, $i = 1, \dots, n$, which applies the subroutine S at most four times to supplementary instances of $\mathcal{I}$, obtained from the original instance by fixing one of the elements in requisitions X^1 and X^n. Note that such a fixing may violate Condition $\mathcal{C}3$. If this happens, the family of requisitions obtained in algorithm $\mathcal{A}$ is transformed into an equivalent one, complying with conditions $\mathcal{C}1 - \mathcal{C}3$. Let us outline the proposed algorithm.

Algorithm $\mathcal{A}$

1. Let $\tilde{f}'$ denote the best found solution to the instance of $\tilde{\mathcal{I}}$ and let $\tilde{\rho}'$ be value of objective function of this solution. Assign initially $\tilde{\rho}' := +\infty$.
2. Perform Steps 2.1-2.2 for each vertex $x \in X^1$:
2.1. Assign $\tilde{X}^1 := \{x\}$, $\tilde{X}^i := X^i$, $i = 2, \dots, n$. Now if $|X^1| = 2$, then the family of requisitions $\tilde{X}^i$, $i = 1, \dots, n$ needs to be transformed to satisfy Condition $\mathcal{C}3$. To this end, an index $j \neq 1$ is found, such that $\tilde{X}^j = \{x, z\}$, and an assignment

$\tilde{X}^j = \{z\}$ is made. Further perform the similar operations with the vertex z etc.
2.2. For each vertex $y \in \tilde{X}^n$ perform Steps 2.2.1-2.2.2:
2.2.1. Assign $\bar{X}^n := \{y\}$, $\bar{X}^i := \tilde{X}^i$, $i = 1, \ldots, n-1$, and if $|\tilde{X}^n| = 2$, then transform the family of requisitions $\bar{X}^i$, $i = 1, \ldots, n$ so that Condition $\mathcal{C}3$ is satisfied, analogously to Step 2.1.
2.2.2. Solve problem $\mathcal{I}$ using Algorithm $\mathcal{S}$. Let f^* be a solution to this problem. If $\rho(f^*) + \rho(\bar{X}^n, \bar{X}^1) < \tilde{\rho}'$, then assign $\tilde{\rho}' := \rho(f^*) + \rho(\bar{X}^n, \bar{X}^1)$ and $\tilde{f}' := f^*$.

It is clear that the solution $\tilde{f}'$ found by algorithm $\mathcal{A}$ will be optimal for problem $\tilde{\mathcal{I}}$. Now since $|X^1| \leqslant 2$, $|X^n| \leqslant 2$, and the transformation of a family of requisitions takes $O(n^2)$ time, so the reduction is polynomially computable. The properties of this reduction imply that problem $\mathcal{I}$ is strongly NP-hard. □

Therefore the following theorem holds.

Theorem 2.2 [16] *The ORP for problem* $1|s_{vu}|C_{\max}$ *is strongly* NP-*hard.*

Although in problem $\mathcal{I}$ we are given a *digraph* G, this problem easily reduces to its modification where G is an ordinary *graph*. This is done by a substitution of each vertex by three vertices (see e.g. [22]) and defining an appropriate family of requisitions X^i, $i = 1, \ldots, n$. Therefore the modification of problem $\mathcal{I}$ on ordinary graphs is also strongly NP-hard and the next result holds.

Theorem 2.3 [16] *The ORP for problem* $1|s_{vu} = s_{uv}|C_{\max}$ *is strongly* NP-*hard.*

2.2 Solving the Optimal Recombination Problem

Given an ORP instance of $1|s_{vu}|C_{\max}$ problem with parent solutions π^1, π^2, one can define an instance of $\mathcal{I}$ as follows.

- Let the number of vertices of digraph G be $n = k$.
- Let each job $v_i \in V$, $i = 1, \ldots, k$, be assigned a vertex $x_i \in X$ of digraph G.
- Let the arc weights be $\rho(x_i, x_j) = s_{v_i, v_j}$ for all $x_i, x_j \in X$, $i \neq j$.
- Let the family of requisitions X^i, $i = 1, \ldots, k$, be such that $X^i = \{\pi^1_i, \pi^2_i\}$ for those i where $\pi^1_i \neq \pi^2_i$ and $X^i = \{\pi^1_i\}$ for the rest of the indices i.

In this case, the set of feasible solutions to problem $\mathcal{I}$ can be mapped to the set of feasible solutions to the ORP for $1|s_{vu}|C_{\max}$ by a bijective mapping so that optimal solutions to problem $\mathcal{I}$ correspond to optimal solutions to the ORP.

An optimal mapping $f^* \in F$ for problem $\mathcal{I}$ can be found in time $O(2^k)$ by enumeration of all sequences π where $\pi_i \in X^i$, $i = 1, \ldots, k$ (feasible as well as infeasible). An obvious modification of the well-known dynamic programming algorithm due to M. Held and R.M. Karp [19] has the same time complexity. It is possible, however, to build a more efficient algorithm for solving problem $\mathcal{I}$, using the approach of A.I. Serdyukov [27] which was developed for estimation of cardinality of the set of feasible solutions to problem $\tilde{\mathcal{I}}$.

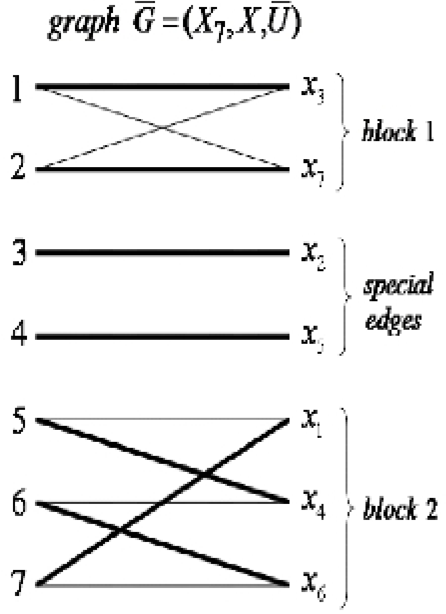

Figure 4: Example of a graph $\bar{G} = (X_7, X, \bar{U})$ with two special edges and two blocks.

Consider a bipartite graph $\bar{G} = (X_k, X, \bar{U})$ defined above. Note that there is a one-to-one correspondence between the set of feasible solutions F to problem $\mathcal{I}$ and the set of perfect matchings $\mathcal{W}$ in graph $\bar{G}$.

An edge $(i, x) \in \bar{U}$ will be called *special*, if (i, x) belongs to all perfect matchings in graph $\bar{G}$. Let us also call the vertices of graph $\bar{G}$ *special*, if they are incident to special edges. A maximal (by inclusion) bi-connected subgraph [7] will be called a *block*. Note that in each block j of graph $\bar{G}$ the degree of any vertex equals two, $j = 1, \ldots, q(\bar{G})$, where $q(\bar{G})$ denotes the number of blocks in graph $\bar{G}$. Then the edges $(i, x) \in \bar{U}$, such that $|X^i| = 1$, are special and belong to none of the blocks, while the edges $(i, x) \in \bar{U}$, such that $|X^i| = 2$, belong to some blocks. Besides that, each block j, $j = 1, \ldots, q(\bar{G})$, of graph $\bar{G}$ contains exactly two maximal (edge disjoint) matchings, so it does not contain the special edges. Hence an edge $(i, x) \in \bar{U}$ is special iff $|X^i| = 1$, and every perfect matching in $\bar{G}$ is defined by a combination of maximal matchings chosen in each of the blocks and the set of all special edges.

As an example consider an instance of $\mathcal{I}$ with $n = k = 7$ and the family of requisitions $X^1 = \{x_3, x_7\}$, $X^2 = \{x_3, x_7\}$, $X^3 = \{x_2\}$, $X^4 = \{x_5\}$, $X^5 = \{x_1, x_4\}$, $X^6 = \{x_4, x_6\}$, $X^7 = \{x_1, x_6\}$. The bipartite graph $\bar{G} = (X_7, X, \bar{U})$ corresponding to this problem is presented in Fig. 4. Here the edges drawn in bold define one maximal matching of a block, and the rest of the edges in the block define another one.

The blocks of graph $\bar{G}$ may be computed in $O(k)$ time, e. g. by means of the "depth first" algorithm [7]. The special edges and maximal matchings in blocks may be found easily in $O(k)$ time.

Therefore, the problem $\mathcal{I}$ is solvable by the following algorithm: Build the bipartite graph $\bar{G}$, identify the set of special edges and blocks and find all maximal matchings in blocks. Enumerate all perfect matchings $W \in \mathcal{W}$ of graph $\bar{G}$ by combining the maximal matchings of blocks and joining them with special edges. Assign the corresponding solution $f \in F$ to each $W \in \mathcal{W}$ and compute $\rho(f)$. As a result one can find $f^* \in F$, such that $\rho(f^*) = \min\limits_{f \in F} \rho(f)$.

Note that $|F| = |\mathcal{W}| = 2^{q(\bar{G})}$, so the time complexity of the above algorithm is $O(k2^{q(\bar{G})})$, where $q(\bar{G}) \leqslant \lfloor \frac{k}{2} \rfloor$ and this bound is tight. Below we propose a modification of this algorithm with time complexity $O(q(\bar{G}) \cdot 2^{q(\bar{G})})$.

Let us carry out some preliminary computations before enumerating all possible combinations of maximal matchings in blocks in order to speed up the evaluation of objective function. We will call a *contact between block j and block $j' \neq j$ (or between block j and a special edge)* the pair of vertices $(i, i+1)$ in the left-hand part of graph $\bar{G}$, such that one of the vertices belongs to the block j and the other one belongs to block j' (or the special edge). A *contact inside a block* will mean a pair of vertices in the left-hand part of a block, if their indices differ exactly by one.

For each block j, $j = 1, \dots, q(\bar{G})$, let us check the presence of contacts inside the block j, between the block j and all special edges, and between the block j and every other block. The time complexity of checking for contacts all vertices in the left-hand part of a block is $O(k)$.

Consider a block j. If a contact $(i, i+1)$ is present inside this block, then each of the two maximal matchings $w^{0,j}$ and $w^{1,j}$ in this block corresponds to an arc of graph G. Also, if block j has a contact to a special edge, each of the two maximal matchings $w^{0,j}$ and $w^{1,j}$ also corresponds to an arc of graph G. For each of the matchings $w^{k,j}$, $k = 0, 1$, let the sum of the weights of arcs corresponding to the contacts inside block j and the contacts to special edges be denoted by P_j^k.

If block j contacts to block j', $j' \neq j$, then each combination of the maximal matchings of these blocks corresponds to an arc of graph G for any contact $(i, i+1)$ between the blocks. If a maximal matching is chosen in each of the blocks, one can sum up the weights of the arcs in G that correspond to all contacts between blocks j and j'. This yields four values which we denote by $P_{jj'}^{(0,0)}$, $P_{jj'}^{(0,1)}$, $P_{jj'}^{(1,0)}$ and $P_{jj'}^{(1,1)}$, where the superscripts identify the matchings chosen in each of the blocks j and j' accordingly.

The above mentioned sums are computed for each block, so the overall time complexity of this pre-processing procedure is $O(k \cdot q(\bar{G}))$.

Now all possible combinations of the maximal matchings in blocks may be enumerated using a Grey code (see e.g. [26]) so that the next combination differs from the previous one by altering a maximal matching only in one of the blocks. Let the binary vector $\delta = (\delta_1, \dots, \delta_{q(\bar{G})})$ define assignments of the maximal matchings in blocks. Namely, $\delta_j = 0$, if the matching $w^{0,j}$ is chosen in block j; otherwise (if the matching $w^{1,j}$ is chosen in block j), we have $\delta_j = 1$. This way every vector δ is bijectively mapped into a feasible solution f_δ to problem $\mathcal{I}$.

In the process of enumeration, a step from the current vector $\bar{\delta}$ to the next vector δ changes the maximal matching in one of the blocks j. The new value of objective function $\rho(f_\delta)$ may be computed via the current value $\rho(f_{\bar{\delta}})$ by the formula $\rho(f_\delta) = \rho(f_{\bar{\delta}}) - P_j^{\bar{\delta}_j} + P_j^{\delta_j} - \sum\limits_{j' \in A(j)} P_{jj'}^{(\bar{\delta}_j, \bar{\delta}_{j'})} + \sum\limits_{j' \in A(j)} P_{jj'}^{(\delta_j, \delta_{j'})}$, where $A(j)$ is the set of blocks contacting to block j. Obviously, $|A(j)| \leqslant q(\bar{G})$, so updating the objective function value for the next solution requires $O(q(\bar{G}))$ time, and the overall time complexity of the modified algorithm for solving Problem $\mathcal{I}$ is $O(q(\bar{G}) \cdot 2^{q(\bar{G})})$.

Therefore, the ORP for $1|s_{vu}|C_{\max}$, as well as Problem $\mathcal{I}$, is solvable in $O(q(\bar{G}) \cdot 2^{q(\bar{G})})$ time. Below it will be shown that for almost all pairs of parent solutions $q(\bar{G}) \leqslant 1.1 \cdot \ln(k)$, i. e. the cardinality of the set of feasible solutions in almost all instances of the ORP for $1|s_{vu}|C_{\max}$ is at most k and these instances are solvable in $O(k \cdot \ln(k))$ time.

Definition 2.4 [27] *A graph $\bar{G} = (X_k, X, \bar{U})$ is called "good" if $q(\bar{G}) \leqslant 1.1 \cdot \ln(k)$; otherwise it is called "bad".*

Definition 2.5 *A pair of parent solutions $\{\pi^1, \pi^2\}$ is called "good" if the graph $\bar{G} = (X_k, X, \bar{U})$ corresponding to these parent solutions is "good"; otherwise the pair $\{\pi^1, \pi^2\}$ is called "bad".*

Note that instead of constant 1.1 in Definition 2.4 one may choose any other constant equal to $1 + \varepsilon$, where $\varepsilon \in (0, \log_2(e) - 1]$. Given such a constant, the ORP has at most k feasible solutions and it is solvable in $O(k\ln(k))$ time.

The following notation will be used below:

- Let $\bar{\Im}_k$ be the set of "good" graphs and let $\tilde{\Im}_k$ denote the set of "bad" graphs.
- Let $\bar{\Re}_k$ be the set of "good" pairs of parent solutions and let $\tilde{\Re}_k$ be the set of "bad" pairs of parent solutions.
- Denote $\Im_k = \bar{\Im}_k \cup \tilde{\Im}_k$, $\Re_k = \bar{\Re}_k \cup \tilde{\Re}_k$.
- Let S_l be the set of permutations of the set $\{1, \dots, l\}$, which do not contain the cycles of length 1.
- Let $\bar{S}_l$ denote the set of permutations from S_l, where the number of cycles is at most $1.1 \cdot \ln(l)$.
- Denote $\tilde{S}_l = S_l \backslash \bar{S}_l$.

The results of A.I. Serdyukov from [27] imply

Proposition 2.6 $|\tilde{S}_l|/|\bar{S}_l| \longrightarrow 0$ *as* $l \to \infty$.

The next theorem is proved by the means of Proposition 2.6.

Theorem 2.7 [16] $|\bar{\Re}_k|/|\Re_k| \longrightarrow 1$ *as* $k \to \infty$.

Proof. The proof consists of two stages: first we estimate the numbers of "good" and "bad" graphs, and after that we estimate the numbers of "good" and "bad" pairs of parent solutions.

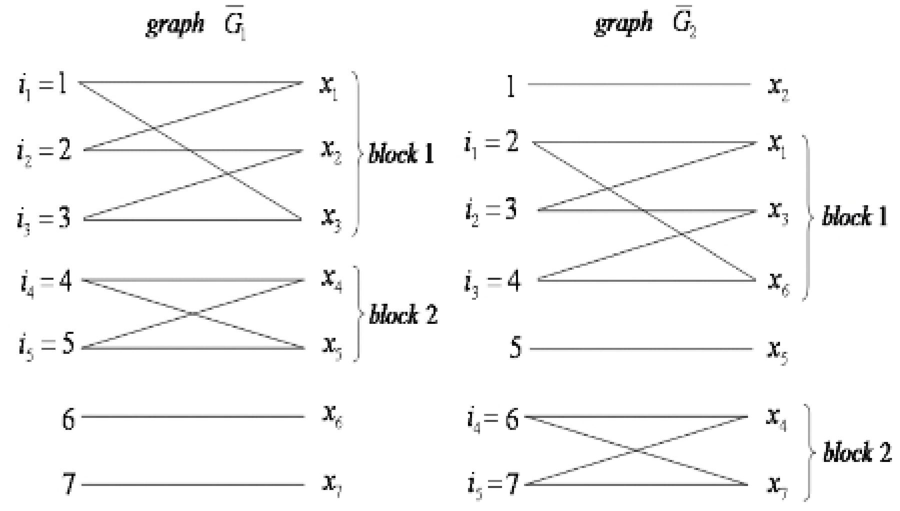

Figure 5: Examples of graphs from class $\Im_7(\sigma)$, where $\sigma = \binom{1\ 2\ 3\ 4\ 5}{2\ 3\ 1\ 5\ 4} \in S_5$.

The values $|\bar{\Im}_k|$ and $|\tilde{\Im}_k|$ may be bounded using the approach from [27]. To this end assign any permutation $\sigma \in S_l$, $l \leqslant k$, a set of bi-partite graphs $\Im_k(\sigma) \subset \Im_k$ as follows. First of all let us assign an arbitrary set of $k-l$ edges to be special. The non-special vertices $\{i_1, i_2, \dots, i_l\} \subset X_k$ of the left-hand part, where $i_j < i_{j+1},\ j = 1, \dots, l-1$, are now partitioned into $\xi(\sigma)$ blocks, where $\xi(\sigma)$ is the number of cycles in permutation σ. Every cycle $(t_1, t_2, \dots, t_r)$ in permutation σ corresponds to some sequence of vertices with indices $\{i_{t_1}, i_{t_2}, \dots, i_{t_r}\}$ belonging to the block associated with this cycle. Finally, it is ensured that for each pair of vertices $\{i_{t_j},\ i_{t_{j+1}}\}$, $j = 1, \dots, r-1$, as well as for the pair $\{i_{t_r},\ i_{t_1}\}$, there exists a vertex in the right-hand part X adjacent to both vertices of the pair.

Consider a permutation $\sigma = \binom{1\ 2\ 3\ 4\ 5}{2\ 3\ 1\ 5\ 4} \in S_5$ with cycles $c_1 = (1,2,3)$ and $c_2 = (4,5)$. Two examples of graphs from class $\Im_7(\sigma)$ are given in Fig. 5. Here block j corresponds to cycle c_j, $j = 1, 2$.

There are $k!$ ways to associate vertices of the left-hand pert to vertices of the right-hand part, therefore the number of different graphs from class $\Im_k(\sigma)$, $\sigma \in S_l$, $l \leqslant k$, is $|\Im_k(\sigma)| = C_k^l \frac{k!}{2^{\xi_1(\sigma)}}$, where $\xi_1(\sigma)$ is the number of cycles of length two in permutation σ. Division by $2^{\xi_1(\sigma)}$ here is due to the fact that for each block that corresponds to a cycle of length two in σ, there are two equivalent ways to number the vertices in its right-hand part.

Let $\sigma = c_1 c_2 \dots c_{\xi(\sigma)}$ be a permutation from set S_l, represented by cycles c_i, $i = 1, \dots, \xi(\sigma)$, and let c_j be an arbitrary cycle of permutation σ of length at least three, $1 \leqslant j \leqslant \xi(\sigma)$. Permutation σ may be transformed into permutation σ^1,

$$\sigma^1 = c_1 c_2 \dots c_{j-1} c_j^{-1} c_{j+1} \dots c_{\xi(\sigma)}, \tag{3}$$

by reversing the cycle c_j. Clearly, permutation σ^1 induces the same subset of graphs in class $\Im_k$ as the permutation σ does. Thus any two permutations σ^1 and σ^2 from set S_l, $l \leqslant k$, induce the same subset of graphs in $\Im_k$, if one of these permutations may be obtained from the other one by several transformations of the form (3). Otherwise the two induced subsets of graphs do not intersect. Besides that $\Im_k(\sigma^1) \cap \Im_k(\sigma^2) = \emptyset$ if $\sigma^1 \in S_{l_1}$, $\sigma^2 \in S_{l_2}$, $l_1 \neq l_2$.

On one hand, if $\sigma \in \bar{S}_l$, $l \leqslant k$, then $\Im_k(\sigma) \subseteq \bar{\Im}_k$. On the other hand, if $\sigma \in \tilde{S}_l$, $l < k$, then either $\Im_k(\sigma) \subseteq \bar{\Im}_k$ or, alternatively, $\Im_k(\sigma) \subseteq \tilde{\Im}_k$ may hold. Therefore,

$$|\bar{\Im}_k| \geqslant \sum_{l=2}^{k} \sum_{\sigma \in \bar{S}_l} C_k^l \frac{k!}{2^{\xi_1(\sigma)} 2^{\xi(\sigma)-\xi_1(\sigma)}} = \sum_{l=2}^{k} \sum_{\sigma \in \bar{S}_l} C_k^l \frac{k!}{2^{\xi(\sigma)}}, \tag{4}$$

$$|\tilde{\Im}_k| \leqslant \sum_{l=\lfloor 1.1\cdot\ln(k)\rfloor}^{k} \sum_{\sigma \in \tilde{S}_l} C_k^l \frac{k!}{2^{\xi_1(\sigma)} 2^{\xi(\sigma)-\xi_1(\sigma)}} = \sum_{l=\lfloor 1.1\cdot\ln(k)\rfloor}^{k} \sum_{\sigma \in \tilde{S}_l} C_k^l \frac{k!}{2^{\xi(\sigma)}}. \tag{5}$$

Now let us estimate the cardinality of sets $\bar{\Re}_k$ and $\tilde{\Re}_k$ to complete the proof. Recall that every graph $\bar{G} \in \Im_k(\sigma)$, $\sigma \in S_l$, $l \leqslant k$ has $\xi(\sigma)$ blocks. The set of edges of any block j, $j = 1, \dots, \xi(\sigma)$, is partitioned into the maximal matchings denoted by $w^j = \{(i_1, x^{i_1}), (i_2, x^{i_2}), \dots, (i_{m_j}, x^{i_{m_j}})\}$ and $\bar{w}^j = \{(i_1, \bar{x}^{i_1}), (i_2, \bar{x}^{i_2}), \dots, (i_{m_j}, \bar{x}^{i_{m_j}})\}$. Then in any instance of the ORP for problem $1|s_{vu}|C_{\max}$, that induces the graph $\bar{G}$, either $\pi^1_{i_m} = x^{i_m}$, $\pi^2_{i_m} = \bar{x}^{i_m}$, $m = 1, \dots, m_j$, or $\pi^1_{i_m} = \bar{x}^{i_m}$, $\pi^2_{i_m} = x^{i_m}$, $m = 1, \dots, m_j$, for all $j = 1, \dots, \xi(\sigma)$. Consequently every bipartite graph from class $\Im_k(\sigma)$ corresponds to $2^{\xi(\sigma)}$ pairs of parent solutions (where pairs $\pi^1 = a$, $\pi^2 = b$ and $\pi^1 = b$, $\pi^2 = a$ are assumed to be different), then in view of (4) and (5) we have:

$$|\bar{\Re}_k| \geqslant \sum_{l=2}^{k} \sum_{\sigma \in \bar{S}_l} C_k^l \frac{k!}{2^{\xi(\sigma)}} 2^{\xi(\sigma)} \geqslant \sum_{l=\lfloor 1.1\cdot\ln(k)\rfloor}^{k} |\bar{S}_l| C_k^l k!, \tag{6}$$

$$|\tilde{\Re}_k| \leqslant \sum_{l=\lfloor 1.1\cdot\ln(k)\rfloor}^{k} \sum_{\sigma \in \tilde{S}_l} C_k^l \frac{k!}{2^{\xi(\sigma)}} 2^{\xi(\sigma)} = \sum_{l=\lfloor 1.1\cdot\ln(k)\rfloor}^{k} |\tilde{S}_l| C_k^l k!. \tag{7}$$

Now assuming $\psi(k) = \max\limits_{l=\lfloor 1.1\cdot\ln(k)\rfloor,\dots,k} |\tilde{S}_l|/|\bar{S}_l|$ and taking into account (6), (7) and Proposition 2.6, we obtain

$$|\tilde{\Re}_k|/|\bar{\Re}_k| \leqslant \psi(k) \to 0 \text{ as } k \to \infty. \tag{8}$$

Finally, the statement of the theorem follows from (8). □

Note that the algorithm proposed for solving the ORP for $1|s_{vu}|C_{\max}$ may be generalized to solve the ORPs for other problems with similar solutions encoding (examples of such problems may be found in [18, 28, 30]). The time complexity of the algorithm in these cases would depend on the time required to evaluate an objective function.

Theorems 2.2 and 2.3 imply NP-hardness of the ORPs for a family of more general scheduling problems, where the number of machines may be greater than 1 and each job may be performed in several modes, using one or more machines, see e.g. [15].

3 FURTHER RESEARCH

In this paper, we did not discuss the population management strategies for the GAs with optimal recombination. It is likely that the general schemes of the GAs and the procedures of parameter adaptation require some revision when the optimal recombination is used. Due to fast localization of the search process in such GAs, it is important to provide a sufficiently large initial population and employ some mechanism for adaptation of the mutation strength (see e.g. [11]). Alternatively, the population management strategy with elitist recombination may be used to avoid the fast localization [30]. Interesting techniques that maintain the diversity of population by constructing the second offspring, as different from the optimal offspring as possible, can be found in [1] and [4].

The overall efficiency of a genetic algorithm depends on finding the trade-off between the time complexity of optimal recombination procedure and the average number of iterations until an optimal solution is found for the first time (the GA runtime). This is another topic for further research. Some results in this area were obtained by means of the experimental analysis [4, 23, 30]. The theoretical methods proposed in [10] and [14] are also applicable in runtime analysis of the GAs with optimal recombination.

One of the ways to control the time complexity of recombination procedure consists in solving the optimal recombination problem approximately. Another way is to apply exact optimal recombination algorithms but not in all occasions. Examples of using these approaches in crossover operators for NP-hard ORPs may be found in [11, 15, 30].

All of the polynomially solvable optimal recombination problems considered above rely upon efficient deterministic algorithms for the Max-Flow / Min-Cut Problem (or the Maximum Matching Problem in the unweighted case). It is an open question, however, whether the fast heuristic methods (see e.g. [3]) may be useful for the polynomially solvable ORPs as well.

Finding approximate solutions to ORP is analogous to finding "first improving" moves in local search algorithms. A formal setting for this approach in the general case of multi-parent recombination is proposed by P. Moscato in [24]. A number of open questions on complexity of recombination problems is posed in [9, 24].

4 CONCLUSION

The results presented in Parts I and II indicate that optimal recombination problem for many NP-hard optimization problems is polynomially solvable, but in many other cases it is NP-hard.

In Part I we have shown that well-known reductions between the NP optimization problems may be useful in development of polynomial-time optimal recombination procedures. Besides that we observed that the choice of solutions encoding has a significant influence upon the complexity of optimal recombination problems. In particular, introduction of additional variables can sometimes simplify the task.

It is natural to expect that average dimensions of the ORP instances might decrease in process of GA execution, as the individuals gain more common genes, so the ORP instances would often have much smaller dimensions, compared to the original problem. In such situations even an NP-hard ORP may turn out to be solvable in practice by the exact methods sufficiently fast. Besides that, the results presented in Part II give evidence that for many NP-hard problems on permutations there might exist faster optimal recombination algorithms, compared to the algorithms known for the original problem.

5 ACKNOWLEDGEMENTS

Partially supported by Russian Foundation for Basic Research grants 12-01-00122 and 13-01-00862 and by Presidium SB RAS (project 7B).

REFERENCES

[1] Aggarwal, C.C., Orlin, J.B. and Tai, R.P., "An optimized crossover for maximum independent set", *Operations Research*, 45 (1997) 225-234.

[2] Ausiello, G., Crescenzi, P., Gambosi, G. et al., *Complexity and approximation: Combinatorial optimization problems and their approximability properties*, Berlin, Springer-Verlag, 1999.

[3] Avis, D., "A survey of heuristics for the weighted matching problem", *Networks*, 13 (1983) 475-493.

[4] Balas, E. and Niehaus, W., "Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems", *Journal of Heuristics*, 4 (2) (1998) 107-122.

[5] Berge, C. *The theory of graphs and its applications,* New York, NY, John Wiley & Sons Inc., 1962.

[6] Cook, W. and Seymour, P., "Tour merging via branch-decomposition", *INFORMS Journal on Computing,* 15 (2) (2003) 233-248.

[7] Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C., *Introduction to Algorithms,* 2nd edition, MIT Press, 2001.

[8] Cotta, C., Alba, E. and 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, Berlin, Springer-Verlag, (1498)1998, 305-314.

[9] Cotta, C., Moscato, P., "The parameterized complexity of multiparent recombination", *Proc. of The 6-th Metaheuristics International Conference,* Vienna, Universität Wien, 2003, 237-241.

[10] Doerr, B., Happ, E. and Klein, C., "Crossover can provably be useful in evolutionary computation" *Theoretical Computer Science* 425 (2012) 17-33.

[11] Dolgui, A., Eremeev, A. and Guschinskaya, O., "MIP-based GRASP and genetic algorithm for balancing transfer lines", *Matheuristics. Hybridizing Metaheuristics and Mathematical Programming,* Ed. by V. Maniezzo, T. Stutzle, and S. Voss, Berlin, Springer-Verlag, 2010, 189-208.

[12] Eppstein, D., "The travelling salesman problem for cubic graphs", *Journal of Graph Algorithms and Applications,* 11 (1) (2007) 61-81.

[13] Eremeev, A.V., "On complexity of optimal recombination for the travelling salesman problem" *Proc. of Evolutionary Computation in Combinatorial Optimization (EvoCOP 2011)*, LNCS, Berlin, Springer Verlag, (6622)2011, 215-225.

[14] Eremeev, A.V., "Non-elitist genetic algorithm as a local search method" *Preprint (arXiv:1307.3463v2 [cs.NE])*, Cornell, Cornell University, 2013, 9 p. URL: http://arxiv.org/abs/1307.3463.

[15] Eremeev, A.V. and Kovalenko, J.V., "On scheduling with technology based machines grouping", *Diskretnyi analys i issledovanie operacii*, 18 (5) (2011) 54-79. (In Russian)

[16] Eremeev, A.V. and Kovalenko, J.V., "On complexity of optimal recombination for one scheduling problem with setup times", *Diskretnyi analys i issledovanie operacii*, 19 (3) (2012) 13-26. (In Russian)

[17] Garey, M. and Johnson, D., *Computers and intractability. A guide to the theory of NP-completeness.* W.H. Freeman and Company, San Francisco, CA, 1979.

[18] Hazir, Ö., Günalay, Y., Erel, E., "Customer order scheduling problem: A comparative metaheuristics study", *International Journal of Advanced Manufacturing Technology*, 37 (2008) 589-598.

[19] Held, M. and Karp, R.M., "A dynamic programming approach to sequencing problems", *SIAM Journal on Applied Mathematics,* 10 (1962) 196-210.

[20] Holland, J., *Adaptation in natural and artificial systems*, Ann Arbor, University of Michigan Press, 1975.

[21] Itai, A., Papadimitriou, C.H. and Szwarcfiter, J.L., "Hamilton paths in grid graphs" *SIAM Journal on Computing,* 11 (4) (1982) 676-686.

[22] Karp, R.M., "Reducibility among combinatorial problems", *Proc. of a Symp. on the Complexity of Computer Computations*, Ed by R.E. Miller and J.W. Thatcher, The IBM Research Symposia Series, New York, NY, Plenum Press, 1972, 85-103.

[23] Kovalenko, J.V., *Complexity of some scheduling problems and evolutionary algorithms of their solution,* Dissertation of Cand. of Sci., Novosibirsk, Sobolev Institute of Mathematics SB RAS, 2013. (In Russian)

[24] Moscato, P. *NP Optimization Problems, approximability and evolutionary computation: from practice to theory*, Ph.D. dissertation, Campinas, University of Campinas, 2001.

[25] Reeves, C.R., "Genetic algorithms for the operations researcher", *INFORMS Journal on Computing,* 9 (3) (1997) 231-250.

[26] Reingold, E.M., Nievergelt, J. and Deo, N. *Combinatorial algorithms: Theory and practice,* Englewood Cliffs, Prentice-Hall, 1977.

[27] Serdyukov, A.I., "On travelling salesman problem with prohibitions", *Upravlaemye systemi*, 17 (1978) 80-86. (In Russian)

[28] Tanaev, V.S., Kovalyov, M.Y. and Shafransky, Y.M. *Scheduling Theory. Group Technologies*, Minsk, Institute of Technical Cybernetics NAN of Belarus, 1998. (In Russian)

[29] Whitley, D., Hains, D. and Howe, A., "A hybrid genetic algorithm for the travelling salesman problem using generalized partition crossover", *Proc. of 11-th Int. Conf. on Parallel Problem Solving from Nature,* LNCS, Berlin, Springer, (6238) 2010, 566-575.

[30] Yagiura, M., Ibaraki, T., "The use of dynamic programming in genetic algorithms for permutation problems", *European Journal of Operational Research*, 92 (1996) 387-401.