Yugoslav Journal of Operations Research
24 (2014) Number 1, 1-20
DOI: 10.2298/YJOR130731040E

Invited Review
OPTIMAL RECOMBINATION IN GENETIC ALGORITHMS FOR COMBINATORIAL OPTIMIZATION PROBLEMS – PART I

Anton V. EREMEEV
Sobolev Institute of Mathematics, Omsk Branch
644099, Omsk, Russia
eremeev@ofim.oscsbras.ru
Julia V. KOVALENKO
Omsk F.M. Dostoevsky State University,
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. We consider efficient reductions of the ORPs, allowing to establish polynomial solvability or NP-hardness of the ORPs, as well as direct proofs of hardness results. Part I presents the basic principles of optimal recombination with a survey of results on Boolean Linear Programming Problems. Part II (to appear in a subsequent issue) is devoted to the ORPs for problems which are naturally formulated in terms of search for an optimal permutation.

Keywords: Genetic Algorithm, Optimal Recombination Problem, complexity, crossover, Boolean Linear Programming
MSC: 90C59, 90C10.

1 INTRODUCTION

The genetic algorithms (GAs) originally suggested by J. Holland [20] are randomized heuristic search methods using an evolving population of sample solutions, based on analogy with the genetic mechanisms in nature. Various modifications of GAs have been widely used in operations research, pattern recognition, artificial intelligence, and other areas (see e.g. [28, 31, 32]). Despite numerous experimental

studies of these algorithms, the theoretical analysis of their efficiency is currently at an early stage [7]. Efficiency of GAs depends significantly on the choice of *crossover* operator, that combines the given *parent* solutions, aiming to produce "good" *offspring* solutions (see e.g. [21]). Originally the crossover operator was proposed as a simple randomized procedure [20], but subsequently the more elaborated problem-specific crossover operators emerged [28].

This paper 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. The ORP is a supplementary problem (usually) of smaller dimension than the original problem, formulated in view of the basic principles of crossover [27].

The first GAs using the optimal recombination appeared in papers of C.C. Agarwal, J.B. Orlin and R.P. Tai [1] and M. Yagiura and T. Ibaraki [31]. These articles provide GAs for the Maximum Independent Set problem and several permutation problems. Subsequent results in [5, 11, 13, 16, 18], and others added more experimental support to expediency of solving the optimal recombination problems in crossover operators.

Interestingly, it turned out that a number of NP-hard optimization problems have efficiently solvable ORPs. The present paper contains a survey of results focused on the issue of efficient solvability vs. intractability of the ORPs.

Part I is structured as follows. The formal definition of the ORP for NP optimization problems is introduced in Section 1. Then, using efficient reductions between the ORPs, it is shown in Section 3 that the optimal recombination is computable in polynomial time for the Maximum Weight Set Packing Problem, the Minimum Weight Set Partition Problem, and for one version of the Simple Plant Location Problem. In Section 3, we also propose an efficient optimal recombination operator for the Boolean Linear Programming Problems with at most two variables per inequality. In Section 4, we consider a number of NP-hard ORPs for the Boolean Linear Programming Problems.

2 OPTIMAL RECOMBINATION IN GENETIC ALGORITHMS

We employ standard definition of an NP optimization problem (see e.g. [3]). By $\{0,1\}^*$, we denote the set of all strings with symbols from $\{0,1\}$ and with arbitrary string length. For a string $S \in \{0,1\}^*$, the symbol $|S|$ denotes its length. The term *polynomial time* stands for the computation time which is upper bounded by a polynomial in length of the input data. Let $\mathbf{R}_+$ denote the set of non-negative reals.

Definition 2.1 *An NP optimization problem* Π *is a triple* $\Pi = (\mathrm{Inst}, \mathrm{Sol}, f_I)$*, where* $\mathrm{Inst} \subseteq \{0,1\}^*$ *is the set of instances of* Π *and:*

1. *The relation* Inst *is computable in polynomial time.*

2. Given an instance $I \in \mathrm{Inst}$, $\mathrm{Sol}(I) \subseteq \{0,1\}^{n(I)}$ *is a set of feasible solutions of* I*, where* $n(I)$ *stands for the dimension of the space of the solutions. Given* $I \in \mathrm{Inst}$ *and* $\mathbf{x} \in \{0,1\}^{n(I)}$*, the decision whether* $\mathbf{x} \in \mathrm{Sol}(I)$ *may be done in polynomial time, and* $n(I) \leq \mathrm{poly}(|I|)$ *for some polynomial* poly*.*

3. Given an instance $I \in \mathrm{Inst}$*,* $f_I : \mathrm{Sol}(I) \to \mathbf{R}_+$ *is the objective function (computable in polynomial time) to be maximized if* Π *is an NP maximization problem or to be minimized if* Π *is an NP minimization problem.*

For the sake of compactness of notation, we will simply put Sol instead of $\mathrm{Sol}(I)$, n instead of $n(I)$ and f instead of f_I, when it is clear which problem instance is implied.

Throughout the paper, we use the term *efficient algorithm* as a synonym for polynomial-time algorithm. A problem solved by such an algorithm is *polynomially solvable.*

Often, it is possible to formulate an NP optimization problem as a Boolean Linear Programming Problem:

$$\max \quad f(\mathbf{x}) = \sum_{j=1}^{n} c_j x_j, \tag{1}$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, \ldots, m, \tag{2}$$

$$x_j \in \{0,1\}, \quad j = 1, \ldots, n. \tag{3}$$

In the context of Boolean Linear Programming Problem, $\mathbf{x} \in \{0,1\}^n$ is treated as a column vector of Boolean variables $x_1, \ldots, x_n$, which belongs to Sol iff the constraints (2) are satisfied.Similar problems where instead of "$\leq$" in (2) stands "$\geq$" or "$=$" for some indices i (or for all i) can be easily transformed to formulation (1)–(3).

Minimization problems can be considered by using the goal function with coefficients c_j of opposite sign. Where appropriate, we will use a more compact notation for problem (1)–(3):

$$\max \{\mathbf{cx} : \mathbf{Ax} \leq \mathbf{b},\ \mathbf{x} \in \{0,1\}^n\},$$

where $\mathbf{A}$ is an $(m \times n)$-matrix with elements a_{ij}, $\mathbf{b} = (b_1, \ldots, b_m)^T$ and $\mathbf{c} = (c_1, \ldots, c_n)$.

2.1 Genetic Algorithms

The simple GA proposed in [20] has been intensively studied and exploited over four decades (see e.g. [29]). This algorithm operates with populations X^t, $t = 1, 2, \ldots$ of binary strings in $\{0,1\}^n$ traditionally called *genotypes*. Each population consists of a fixed number of genotypes N, which is assumed to be even. In a *selection operator* Sel, each parent is drawn from the previous population X^t independently with probability distribution assigning each genotype a probability

proportional to its *fitness*, where fitness is measured by the value of the objective function or a composition of the objective function with some monotonic function.

A pair of offspring genotypes is created through recombination and mutation stages. In the recombination stage, a crossover operator Cross exchanges random substrings between pairs of parent genotypes ξ, η with a given constant probability $P_{\rm c}$ so that

$$\mathbf{P}\left\{\xi' = (\xi_1, ..., \xi_j, \eta_{j+1}, ..., \eta_n),\ \eta' = (\eta_1, ..., \eta_j, \xi_{j+1}, ..., \xi_n)\right\} = \frac{P_{\rm c}}{n-1},\ j = 1, ..., n-1,$$

$$\mathbf{P}\{\xi' = \xi,\ \eta' = \eta\} = 1 - P_{\rm c}.$$

In the *mutation operator* Mut, each bit of an offspring genotype may be flipped with a constant mutation probability $P_{\rm m}$, which is usually chosen to be relatively small. When the whole population X^{t+1} of N offspring is constructed, the GA proceeds to the next iteration $t+1$. An initial population X^0 is generated randomly with independent choice of all bits in genotypes.

A plenty of variants of GA have been developed since the publication of the simple GA in [20], sharing the basic ideas, but using different population management strategies, selection, crossover and mutation operators [29]. The practice shows that the best results are obtained when the GAs are designed in view of the specific features of the optimization problem to be solved. A number of such problem-specific GAs make use of crossover operators that find exact or at least approximate solution to the optimal recombination problem.

2.2 Formulation of Optimal Recombination Problem

In this paper, the ORPs are considered assuming that a binary representation of solutions in genotypes is identical to the solutions encoding of the NP optimization problem. Besides, it is assumed that X^0 consists of feasible solutions and operators Cross and Mut maintain feasibility of solutions, i. e. $\text{Cross} : \text{Sol}^2 \to \text{Sol}^2$, $\text{Mut} : \text{Sol} \to \text{Sol}$. Therefore, the term "genotype" will denote an element of the set of feasible solutions Sol.

Note that there may be a number of NP optimization problems which essentially correspond to the same problem in practice. Such formulations are usually easy to transform to each other, but the solution representations may be quite different in the degree of degeneracy, the number of local optima for some standard neighborhood definitions, the length of encoding strings and other parameters important for heuristic algorithms. Since the method of solutions representation is crucial for recombination operators, in what follows, we always explicitly indicate which solutions encoding is used in formulation of an NP optimization problem.

In general, an instance of an NP optimization problem may have no feasible solutions. However, w.r.t. the optimal recombination problem such cases are not meaningful, since there exist no feasible parent solutions. Therefore, in the context of optimal recombination below, we will always assume that $\text{Sol} \neq \emptyset$.

The following definition of optimal recombination problem is motivated by the principles of *(strictly) gene transmitting* recombination formulated by N. Radcliffe [27].

Definition 2.2 *Given an NP optimization problem* $\Pi = (\mathrm{Inst}, \mathrm{Sol}, f)$*, the optimal recombination problem for* Π *is the NP optimization problem* $\overline{\Pi} = (\overline{\mathrm{Inst}}, \overline{\mathrm{Sol}}, \overline{f})$*, where for every instance* $\overline{I} = (I, \mathbf{p}^1, \mathbf{p}^2) \in \overline{\mathrm{Inst}}$ *holds*
$I \in \mathrm{Inst}$*,* $\mathbf{p}^1 = (p^1_1, \ldots, p^1_{n(I)}) \in \mathrm{Sol}(I)$*,* $\mathbf{p}^2 = (p^2_1, \ldots, p^2_{n(I)}) \in \mathrm{Sol}(I)$*, and it is assumed that*

$$\overline{\mathrm{Sol}}(\overline{I}) = \{\mathbf{x} \in \mathrm{Sol}(I) |\ x_j = p^1_j \text{ or } x_j = p^2_j,\ j = 1, \ldots, n(I)\}. \tag{4}$$

The optimization criterion in $\overline{I}$ *is the same as in* I*, i. e.* $\overline{f}_{\overline{I}} \equiv f_I$*.*

Feasible solutions $\mathbf{p}^1, \mathbf{p}^2$ to problem I are called *parent* solutions for the problem $\overline{I} = (I, \mathbf{p}^1, \mathbf{p}^2)$. In what follows, we denote the set of coordinates, where the parent solutions have different values, by $D(\mathbf{p}^1, \mathbf{p}^2) = \{j : p^1_j \neq p^2_j\}$. These are the variables subject to optimization in the ORP. All other variables are "fixed" in the ORP, being equal to the values of the corresponding coordinates in the parent solutions.

Other formulations of recombination subproblem, which may be found in literature, are the examples of *allelic dynastically optimal recombination* [10]. In particular, in [8, 9, 14, 25] promising experimental results are demonstrated by GAs where the recombination subproblem is defined by "fixing" only those genes, where both parent genotypes contain zeros.

3 EFFICIENTLY SOLVABLE OPTIMAL RECOMBINATION PROBLEMS

As the first examples of efficiently solvable ORPs, we will consider the following three well-known problems. Given a graph $G = (V, E)$ with vertex weights $w(v)$, $v \in V$,

- the Maximum Weight Independent Set Problem asks for a subset $S \subseteq V$, such that each edge $e \in E$ has at least one endpoint outside S (i.e. S is an independent set) and the weight $\sum_{v \in S} w(v)$ of S is maximized;
- the Maximum Weight Clique Problem asks for a maximum weight subset $Q \subseteq V$, such that any two vertices u, v in Q are adjacent (i. e. Q is a clique);
- the Minimum Weight Vertex Cover Problem asks for a minimum weight subset $C \subseteq V$, such that any edge $e \in E$ is incident at least to one of the vertices in C (i. e. C is a vertex cover).

Suppose, the vertices of graph G are ordered. We will consider these three problems using the standard binary representation of solutions by the indicator vectors, assuming $n = |V|$ and $x_j = 1$ iff vertex v_j belongs to the subset represented by $\mathbf{x}$. The following result is due to E. Balas and W. Niehaus.

Theorem 3.1 [4] *The ORP for the Maximum Weight Clique Problem is solvable in time* $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3 + n)$*.*

Proof. Consider the Maximum Weight Clique Problem on a given graph G with two parent cliques Q_1 and Q_2, represented by binary vectors $\mathbf{p}^1$ and $\mathbf{p}^2$. An offspring solution Q should contain the whole set of vertices $Q_1 \cap Q_2$, besides that Q should not contain the elements from the set $V \setminus (Q_1 \cup Q_2)$, while the vertices with indices from the set $D(\mathbf{p}^1, \mathbf{p}^2)$ should be chosen optimally. The latter task can be formulated as a Maximum Weight Clique Problem in subgraph $H = (V', E')$, which is induced by the subset of vertices with indices from $D(\mathbf{p}^1, \mathbf{p}^2)$. To find a clique of maximum weight in H, it is sufficient to find a minimum weight vertex cover C' in the complement graph $\bar{H}$ and take $V' \setminus C'$. Note that $\bar{H}$ is a bipartite graph, so let V'_1, V'_2 be the subsets of vertices in this bipartition.

The Minimum Weight Vertex Cover C' for $\bar{H}$ can be found by solving the s-t-Minimum Cut Problem on a supplementary network $\mathcal{N}$, based on $\bar{H}$, as described e.g. in [19]: in this network, an additional vertex s is connected by outgoing arcs with the vertices of set V'_1, and the other additional vertex t is connected by incoming arcs to the subset V'_2. The capacities of the new arcs are equal to the weights of the adjacent vertices in $\bar{H}$. Each edge of $\bar{H}$ is viewed as an arc, directed from its endpoint $u \in V'_1$ to the endpoint $v \in V'_2$. The arc capacity is set to $\max\{w(u), w(v)\}$. This s-t-Minimum Cut Problem can be solved in $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3)$ time using the maximum-flow algorithm due to A.V. Karzanov – see e.g. [26].

We will assume that the s-t-minimum cut contains only the arcs outgoing from s or incoming into t, because if some arc (u, v), $u \in V'_1, v \in V'_2$ enters the s-t-minimum cut, one can substitute it by (s, u) or (v, t), which does not increase the weight of the cut.

Finally, it is easy to verify that $(V'_1 \cup V'_2) \setminus C'$ joined with $Q_1 \cap Q_2$ defines the required ORP solution.

Since the parent solutions are given by the n-dimensional indicator vectors $\mathbf{p}^1$ and $\mathbf{p}^2$, we get the overall time complexity $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3 + n)$. □

Note that if all vertex weights are equal, then the time complexity of Karzanov's algorithm for the networks of simple structure (as the one constructed in the proof of Theorem 3.1) reduces to $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^{2.5})$ – see [26]. The same time complexity has the Optimized Crossover Algorithm [1, 5] for solving this ORP in the unweighted case. The algorithm is based on reduction of the ORP to the Maximum Matching problem.

The Maximum Weight Independent Set and the Minimum Weight Vertex Cover Problems are closely related to the Maximum Weight Clique Problem (see e.g. [17]). It is sufficient to consider the complement graph and to change the optimization criterion accordingly. Then there is a bijection between the set of feasible solutions of each of these problems and the set of feasible solutions of the corresponding Maximum Weight Clique Problem. In the case of Maximum Weight Independent Set, the bijection is an identity mapping, while in the case of the Minimum Weight Vertex Cover, the bijection alters each bit in $\mathbf{x}$. In the first case, the mapped feasible solutions retain their objective function values, while in the second case the original objective function values are subtracted from the weight of all vertices. In view of these relationships, Theorem 3.1 implies that the ORPs for the Maximum Weight Independent Set and the Minimum Weight Vertex Cover Problems are solvable in time $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3 + n)$ as well. Indeed, it suffices to consider the corresponding instance of the ORP for the Maximum Clique Problem,

solve this ORP in $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3 + n)$ time and map the obtained solution back into the set of feasible solutions of the original problem.

The above arguments illustrate that when one NP-optimization problem transforms efficiently to another one, the corresponding ORPs may reduce efficiently as well. The following subsection is devoted to analysis of the situations where such arguments apply.

3.1 Reductions of Optimal Recombination Problems

The usual approach to spreading a class of polynomially solvable (or intractable) problems consists in building chains of efficient problem reductions. In order to apply this approach to optimal recombination problems, we shall first formulate a relatively general reducibility condition for NP optimization problems.

Proposition 3.2 *Let* $\Pi_1 = (\mathrm{Inst}_1, \mathrm{Sol}_1, f_I)$ *and* $\Pi_2 = (\mathrm{Inst}_2, \mathrm{Sol}_2, g_{I'})$ *be NP optimization problems with maximization (minimization) criteria, and there exists a mapping* $\alpha : \mathrm{Inst}_1 \to \mathrm{Inst}_2$ *and an injective mapping* $\beta : \mathrm{Sol}_1(I) \to \mathrm{Sol}_2(\alpha(I))$, *such that given* $I \in \mathrm{Inst}_1$,

1. *for any* $\mathbf{x}, \mathbf{x}' \in \mathrm{Sol}_1(I)$, *satisfying the condition*

$$f_I(\mathbf{x}) > f_I(\mathbf{x}'), \tag{5}$$

the following inequality holds

$$g_{\alpha(I)}(\beta(\mathbf{x})) > g_{\alpha(I)}(\beta(\mathbf{x}')) \tag{6}$$

(if Π_1 *is a minimization problem, the inequality sign in (5) changes into "<"; if* Π_2 *is a minimization problem, the inequality sign in (6) changes into "<");*

2. *if* $\mathbf{y} \in \beta(\mathrm{Sol}_1(I))$, $\mathbf{y}' \in \mathrm{Sol}_2(\alpha(I))$, *and*

$$g_{\alpha(I)}(\mathbf{y}') \geq g_{\alpha(I)}(\mathbf{y}), \tag{7}$$

then $\mathbf{y}' \in \beta(\mathrm{Sol}_1(I))$ *(if* Π_2 *is a minimization problem, the inequality sign in (7) changes into "*$\leq$*").*

Then Π_1 *transforms to* Π_2, *so that any instance* $I \in \mathrm{Inst}_1$ *can be solved in time* $O(T_\alpha(I) + T_{\beta^{-1}}(I) + T(I))$, *where* $T_\alpha(I)$ *is the computation time of* $\alpha(I)$; $T_{\beta^{-1}}(I)$ *is an upper bound on the computation time of* $\beta^{-1}(\mathbf{y})$, $\mathbf{y} \in \beta(\mathrm{Sol}_1(I))$; $T(I)$ *is the time complexity of solving the problem* $\alpha(I)$.

Proof. Suppose $I \in \mathrm{Inst}_1$ and consider an optimal solution $\mathbf{y}^*$ to problem $\alpha(I)$. According to condition 2, if $\mathrm{Sol}_1(I) \neq \emptyset$, then $\mathbf{y}^* \in \beta(\mathrm{Sol}_1(I))$. By proof from the contrary, in view of condition 1, we conclude that if $\mathrm{Sol}_1(I) \neq \emptyset$, then $\beta^{-1}(\mathbf{y}^*)$ is an optimal solution to I. □

Note that condition 2 in Proposition 3.2 implies that the set of feasible solutions of problem Π_1 is mapped into a set of "sufficiently good" feasible solutions to Π_2 (in terms of objective function). This property is observed in many transformations, involving penalization of "undesired" solutions to Π_2 (see e.g. [6, 24]).

If the computation times $T_\alpha(I)$ and $T_{\beta^{-1}}(I)$ are polynomially bounded w.r.t. $|I|$, then Proposition 3.2 provides a sufficient condition of polynomial reducibility of one NP optimization to another.

The following proposition is aimed at obtaining efficient reductions of one ORP to another, when there exist efficient transformations between the corresponding NP optimization problems.

Proposition 3.3 *Let* $\Pi_1 = (\mathrm{Inst}_1, \mathrm{Sol}_1, f_I)$ *and* $\Pi_2 = (\mathrm{Inst}_2, \mathrm{Sol}_2, g_{I'})$ *be both NP optimization problems, where* $\mathrm{Sol}_1(I) \subseteq \{0,1\}^{n_1(I)}$, $\mathrm{Sol}_2(I') \subseteq \{0,1\}^{n_2(I')}$ *and there exist the mappings* α *and* β *for which the condition of Proposition 3.2 holds, and besides that:*

(i) For any $j = 1, \ldots, n_1(I)$ *there exists such* $k(j)$ *that* $\beta^{-1}(\mathbf{y})_j$ *is a function of* $y_{k(j)}$*, when* $\mathbf{y} = (y_1, \ldots, y_{n_2}) \in \beta(\mathrm{Sol}_1(I))$.

(ii) For any $k = 1, \ldots, n_2(\alpha(I))$ *there exists such* $j(k)$ *that* $\beta(\mathbf{x})_k$ *is a function of* $x_{j(k)}$*, when* $\mathbf{x} = (x_1, \ldots, x_{n_1}) \in \mathrm{Sol}_1(I)$.

Then $\overline{\Pi}_1$ *reduces to* $\overline{\Pi}_2$*, and any instance* $\overline{I} = (I, \mathbf{p}^1, \mathbf{p}^2)$ *from ORP* $\overline{\Pi}_1$ *is solvable in time* $O(T_\alpha(I) + T_\beta(I) + T_{\beta^{-1}}(I) + \overline{T}(I, \mathbf{p}^1, \mathbf{p}^2))$*, where* $\overline{T}(I, \mathbf{p}^1, \mathbf{p}^2)$ *is the time complexity of solving ORP* $(\alpha(I), \beta(\mathbf{p}^1), \beta(\mathbf{p}^2))$*, and* $T_\beta(I)$ *is an upper bound on computation time of* $\beta(\mathbf{x})$, $\mathbf{x} \in \mathrm{Sol}_1(I)$.

Proof. Without loss of generality, we shall assume that Π_1 and Π_2 are maximization problems. Suppose that an instance I of problem Π_1 and two parent solutions $\mathbf{p}^1, \mathbf{p}^2 \in \mathrm{Sol}_1(I)$ are given. These solutions correspond to feasible solutions $\mathbf{q}^1 = \beta(\mathbf{p}^1)$, $\mathbf{q}^2 = \beta(\mathbf{p}^2)$ to problem $\alpha(I)$.

Now let us consider the ORP for instance $\alpha(I)$ of Π_2 with parent solutions $\mathbf{q}^1, \mathbf{q}^2$. The optimal solution to this ORP $\mathbf{y}' \in \mathrm{Sol}_2(\alpha(I))$ can be transformed in time $T_{\beta^{-1}}$ into a feasible solution $\mathbf{z} = \beta^{-1}(\mathbf{y}') \in \mathrm{Sol}_1(I)$.

Note that for all $j \notin D(\mathbf{p}^1, \mathbf{p}^2)$ hold $z_j = p^1_j = p^2_j$. Indeed, by condition (i), for any $j = 1, \ldots, n_1(I)$, there exists such $k(j)$ that

(I) either $\beta^{-1}(\mathbf{y})_j = y_{k(j)}$ for all $\mathbf{y} \in \beta(\mathrm{Sol}_1(I))$, or

(II) $\beta^{-1}(\mathbf{y})_j = 1 - y_{k(j)}$ for all $\mathbf{y} \in \beta(\mathrm{Sol}_1(I))$, or

(III) $\beta^{-1}(\mathbf{y})_j$ is constant on $\beta(\mathrm{Sol}_1(I))$.

In the case (I) for all $j \notin D(\mathbf{p}^1, \mathbf{p}^2)$, we have $z_j = y'_{k(j)}$. Now $y'_{k(j)} = q^1_{k(j)}$ by the definition of the ORP, since $q^1_{k(j)} = p^1_j = p^2_j = q^2_{k(j)}$. So, $z_j = q^1_{k(j)} = p^1_j = p^2_j$. The case (II) is treated analogously. Finally, the case (III) is trivial since $\mathbf{z}, \mathbf{p}^1, \mathbf{p}^2 \in \beta^{-1}(\beta(\mathrm{Sol}_1(I)))$. So, $\mathbf{z}$ is a feasible solution to the ORP for Π_1.

To prove the optimality of $\mathbf{z}$ for the instance $\overline{I}$ from the ORP $\overline{\Pi}_1$, we will assume by contradiction that there exists a feasible solution $\mathbf{z}' = (z'_1, \ldots, z'_{n_1}) \in \mathrm{Sol}_1(I)$ such that $\mathbf{z}'_j = p^1_j = p^2_j$ for all $j \notin D(\mathbf{p}^1, \mathbf{p}^2)$, and $f_I(\mathbf{z}') > f_I(\mathbf{z}')$. Then $g_{\alpha(I)}(\beta(\mathbf{z}')) > g_{\alpha(I)}(\beta(\mathbf{z})) = g_{\alpha(I)}(\mathbf{y}')$. But $\beta(\mathbf{z}')$ coincides with $\mathbf{q}^1$ and $\mathbf{q}^2$ in all coordinates $k \notin D(\mathbf{q}^1, \mathbf{q}^2)$ according to condition (ii) (it is sufficient to consider three cases similar to (I) – (III) in order to verify this). Thus $\mathbf{y}'$ is not an optimal solution to the ORP for $\alpha(I)$, which is a contradiction. □

The special case of this proposition where $n_1(I) \equiv n_2(I')$ and $k(j) \equiv j$, $j(k) \equiv k$ appears to be the most applicable, as it is demonstrated in what follows.

Let us use Proposition 3.3 to obtain an efficient optimal recombination algorithm for the *Maximum Weight Set Packing Problem:*

$$\max\{f_{\text{pack}}(\mathbf{x}) = \mathbf{cx} : \mathbf{Ax} \leq \mathbf{e}, \mathbf{x} \in \{0,1\}^n\}, \tag{8}$$

where $\mathbf{A}$ is a given $(m \times n)$-matrix of zeros and ones. Here and below $\mathbf{e}$ is an m-dimensional column vector of ones. The transformation α from the Set Packing to the Maximum Weight Independent Set Problem (with the standard binary solutions encoding) consists in building a graph on a set of vertices $v_1, \ldots, v_n$ with weights $c_1, \ldots, c_n$. Each pair of vertices v_j, v_k is connected by an edge iff j and k both belong at least to one of the subsets $N_i = \{j : a_{ij} \neq 0\}$. In this case, β is an identical mapping. Application of Proposition 3.3 leads to

Corollary 3.4 [15] *The ORP for the Maximum Weight Set Packing Problem* (8) *is solvable in time* $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3 + n^2 m)$.

Now, we can prove the polynomial solvability of the next two problems in Boolean linear programming formulations.

The first problem is the *Minimum Weight Set Partition Problem:*

$$\min\{f_{\text{part}}(\mathbf{x}) = \mathbf{cx} : \mathbf{Ax} = \mathbf{e}, \mathbf{x} \in \{0,1\}^n\}, \tag{9}$$

where $\mathbf{A}$ is a given $(m \times n)$-matrix of zeros and ones.

The second problem is the *Simple Plant Location Problem.* Suppose that there are n sites of potential facility location for production of some uniform product. The cost of opening a facility at a location i is $C_i \geq 0$. Each opened facility can provide an unlimited amount of commodity.

Suppose that there are m customers who require service, and that the cost of serving a client j by facility i is $c_{ij} \geq 0$. The goal is to determine a set of sites where the facilities should be opened, so as to minimize the total opening and service cost. This problem can be formulated as a nonlinear Boolean Programming Problem:

$$\min \quad f_{\text{splp}}(\mathbf{x}) = \sum_{i=1}^{n} C_i x_i + \sum_{j=1}^{m} \min_{i:x_i=1} c_{ij}, \tag{10}$$

s. t.

$$\sum_{i=1}^{n} x_i \geq 1. \tag{11}$$

Here, the vector of variables $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ is an indicator vector for the set of opened facilities. Note that given a vector of opened facilities, a least cost assignment of clients to these facilities is easy to find. An optimal solution to the Simple Plant Location Problem in the above formulation is denoted by $\mathbf{x}^*$.

The Simple Plant Location Problem is strongly NP-hard even if the matrix (c_{ij}) satisfies the triangle inequality [23]. Interconnections of this problem to other well-known optimization problems may be found in [6, 24] and the references provided there.

Alternatively, the Simple Plant Location Problem may be formulated as a Boolean Linear Programming Problem:

$$\min \quad f_{\text{splp}}(\mathbf{Y}, \mathbf{u}) = \sum_{k=1}^{K} \sum_{\ell=1}^{L} c_{k\ell} y_{k\ell} + \sum_{k=1}^{K} C_k u_k, \tag{12}$$

$$\sum_{k=1}^{K} y_{k\ell} = 1, \quad \ell = 1, \ldots, L, \tag{13}$$

$$u_k \geq y_{k\ell}, \quad k = 1, \ldots, K, \ \ell = 1, \ldots, L, \tag{14}$$

$$y_{k\ell} \in \{0, 1\}, \ u_k \in \{0, 1\}, \quad k = 1, \ldots, K, \ \ell = 1, \ldots, L. \tag{15}$$

Here and below, we denote the $(K \times L)$-matrix of Boolean variables $y_{k\ell}$ by $\mathbf{Y}$, and the K-dimensional vector of Boolean variables u_k is denoted by $\mathbf{u}$. This formulation of the Simple Plant Location Problem is equivalent to (10) – (11). However, according to Definition 2.1, the NP optimization problem (10)–(11) is different from the problem (12)–(15)since in the first case, the feasible solutions are encoded by vectors $\mathbf{x} \in \{0, 1\}^n$ while in the second case, the feasible solutions are encoded by pairs $(\mathbf{Y}, \mathbf{u})$.

On one hand, in Section 4 it will be shown that the ORP for the Simple Plant Location Problem (10)–(11) is NP-hard. On the other hand, the following corollary shows that the ORP for Simple Plant Location Problem (12)–(15) is efficiently solvable, as well as the ORP for the Set Partition Problem (9).

Corollary 3.5 [15]

(i) The ORP for the Minimum Weight Set Partition Problem (9) is solvable in time $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3 + n^2 m)$.

(ii) The ORP for the Simple Plant Location Problem in Boolean Linear Programming formulation (12)–(15) is solvable in polynomial time.

Proof. For both cases, we will use well-known transformations of the corresponding NP optimization problems to the Minimum Weight Set Packing Problem (see e.g. the transformations T2 and T5 in [24]).

(i) Let us denote the Minimum Weight Set Partition Problem by Π_1, and let the Set Packing Problem be Π_2. Since $\text{Sol}_1(I) \neq \emptyset$, the problem I is equivalent to

$$\min \quad \sum_{j=1}^{n} c_j x_j + \lambda \sum_{i=1}^{m} w_i,$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j + w_i = 1, \ i = 1, \ldots, m,$$

$$x_j \in \{0, 1\}, \ j = 1, \ldots, n; \quad w_i \geq 0, \ i = 1, \ldots, m,$$

where $\lambda > 2 \sum_{j=1}^{n} |c_j|$ is a penalty factor assuring that all "artificial" slack variables w_i become zeros in the optimal solution. By substitution of w_i into the objective function, the latter model transforms into

$$\min \left\{ \lambda m + \sum_{j=1}^{n} \left(c_j - \lambda \sum_{i=1}^{m} a_{ij} \right) x_j : \mathbf{A}\mathbf{x} \leq \varepsilon, \ \mathbf{x} \in \{0, 1\}^n \right\},$$

which is equivalent to the following instance $\alpha(I)$ of the Set Packing Problem Π_2:

$$\max \left\{ g(\mathbf{x}) = \sum_{j=1}^{n} \left(\lambda \sum_{i=1}^{m} a_{ij} - c_j \right) x_j : \mathbf{A}\mathbf{x} \leq \varepsilon,\ \mathbf{x} \in \{0,1\}^n \right\}.$$

Assume that β is an identical mapping. Then, each feasible solution $\mathbf{x}$ of the Set Partition Problem is a feasible solution to problem Π_2 with the objective function value $g(\mathbf{x}) = \lambda m - f_{\text{part}}(\mathbf{x}) > \lambda(m - 1/2)$. At the same time, if a vector $\mathbf{x}'$ is feasible for problem Π_2 but infeasible for Π_1, it will have the objective function value $g(\mathbf{x}') = \lambda(m-k) - f_{\text{part}}(\mathbf{x}')$, where k is the number of constraints of the form $\sum_{j=1}^{n} a_{ij}x_j = 1$, which are violated by $\mathbf{x}'$. So, β is a bijection from $\text{Sol}_1(I)$ to a set of feasible solutions with sufficiently high values of the objective function:

$$\{\mathbf{x} \in \text{Sol}_2(\alpha(I)) \mid g(\mathbf{x}) > \lambda(m - 1/2)\}.$$

The complexity of ORP for Π_2 is bounded by Corollary 3.4. Thus, application of Proposition 3.3 completes the proof of part (i).

(ii) Let Π_1' be the Simple Plant Location Problem. Analogously to the case (i), we will convert equations (13) into inequalities. To this end, we rewrite (13) as $\sum_{k=1}^{K} y_{k\ell} + w_\ell = 1,\ \ell = 1, \ldots, L$, with nonnegative slack variables w_ℓ and ensure that all of them turn into zero in the optimal solution, by means of a penalty term $\lambda \sum_{\ell=1}^{L} w_\ell$ added to the objective function. Here

$$\lambda > \sum_{k=1}^{K} C_k + \sum_{\ell=1}^{L} \max_{k=1,\ldots,K} c_{k\ell}.$$

Eliminating variables w_ℓ, we substitute (13) by $\sum_{k=1}^{K} y_{k\ell} \leq 1,\ \ell = 1, \ldots, L$, and change the penalty term into $\lambda L - \lambda \sum_{\ell=1}^{L} \sum_{k=1}^{K} y_{k\ell}$. Multiplying the criterion by -1, and introducing a new set of variables $\overline{u}_k = 1 - u_k,\ k = 1, \ldots, K$, we obtain the following NP maximization problem Π_2':

$$\max\ \ g'(\mathbf{Y}, \overline{\mathbf{u}}) = \sum_{k=1}^{K} \sum_{\ell=1}^{L} (\lambda - c_{k\ell}) y_{k\ell} + \sum_{k=1}^{K} C_k \overline{u}_k - \lambda L - \sum_{k=1}^{K} C_k, \tag{16}$$

subject to

$$\sum_{k=1}^{K} y_{k\ell} \leq 1, \quad \ell = 1, \ldots, L, \tag{17}$$

$$\overline{u}_k + y_{k\ell} \leq 1, \quad k = 1, \ldots, K,\ \ell = 1, \ldots, L, \tag{18}$$

$$y_{k\ell} \in \{0,1\},\ \overline{u}_k \in \{0,1\}, \quad k = 1, \ldots, K,\ \ell = 1, \ldots, L, \tag{19}$$

where $\overline{\mathbf{u}} = (\overline{u}_1, \ldots, \overline{u}_K)$. Obviously, Π_2' is a special case of the Set Packing Problem, up to an additive constant $-\lambda L - \sum_{k=1}^{K} C_k$ in the objective function. Thus, we have defined the mapping $\alpha(I)$.

Assume that β maps identically all variables $y_{k\ell}$ and transforms the variables u_k into $\overline{u}_k = 1 - u_k,\ k = 1, \ldots, K$. Then, each feasible solution $(\mathbf{Y}, \mathbf{u})$ of

the Simple Plant Location Problem is mapped into a feasible solution to problem Π'_2 with an objective function value $g'(\mathbf{Y}, \overline{\mathbf{u}}) = -f_{\text{splp}}(\mathbf{Y}, \mathbf{u}) > -\lambda$. If a pair $(\mathbf{Y}, \overline{\mathbf{u}})$ is feasible for problem Π'_2 but $(\mathbf{Y}, \mathbf{u})$ is infeasible in Π'_1 then, $g'(\mathbf{Y}, \overline{\mathbf{u}}) \leq -f_{\text{splp}}(\mathbf{Y}, \mathbf{u}) - \lambda$, because at least one of the equalities (13) is violated by $(\mathbf{Y}, \mathbf{u})$.

The ORP for the problem Π'_2 can be solved in polynomial time by Corollary 3.4, thus Proposition 3.3 gives the required optimal recombination algorithm for Π'_1. □

3.2 Boolean Linear Programming Problems and Hypergraphs

The starting point of all reductions considered above was Theorem 3.1 which may be viewed as an efficient reduction of the ORP for the Maximum Weight Clique Problem to the Maximum Weight Independent Set Problem in a bipartite graph. In order to generalize this approach, now we will move from bipartite graphs to 2-colorable hypergraphs.

A hypergraph $H = (V, E)$ is given by a finite nonempty set of vertices V and a set of edges E, where each edge $e \in E$ is a subset of V. A subset $S \subseteq V$ is called *independent* if none of the edges $e \in E$ is a subset of S. The Maximum Weight Independent Set Problem on hypergraph $H = (V, E)$ with rational vertex weights $w(v),\ v \in V$ asks for an independent set S with maximum weight $w(S) = \sum_{v \in S} w(v)$.

A generalization of the bipartite graph is the *2-colorable hypergraph*: there exists a partition of the vertex set V into two disjoint independent subsets C_1 and C_2. The partition $V = C_1 \cup C_2$, $C_1 \cap C_2 = \emptyset$ is called a 2-coloring of H and C_1, C_2 are the color classes.

Let us denote by N_i the set of indices of non-zero elements in constraint i of the Boolean Linear Programming Problem (1)-(3). In the sequel, we will assume that at least one of the subsets N_i contains two or more elements (otherwise the problem is solved trivially).

Theorem 3.6 [15] *The ORP for Boolean Linear Programming Problem (1)-(3) reduces to the Maximum Weight Independent Set Problem on a 2-colorable hypergraph with a 2-coloring given in the input. Each edge in the 2-colorable hypergraph contains at most $N_{\max}$ vertices, where $N_{\max} = \max_{i=1,\ldots,m} |N_i|$, and the time complexity of this reduction is $O(m(2^{N_{\max}} + n))$.*

Proof. Given an instance of the Boolean Linear Programming Problem with parent solutions $\mathbf{p}^1$ and $\mathbf{p}^2$, let us denote $|D(\mathbf{p}^1, \mathbf{p}^2)|$ by d and construct a hypergraph H on $2d$ vertices, assigning each variable $x_j,\ j \in D(\mathbf{p}^1, \mathbf{p}^2)$, a couple of vertices v_j and v_{n+j}. In order to model each of the linear constraints for $i = 1, \ldots, m$, we will look through all possible combinations of the Boolean variables from $D(\mathbf{p}^1, \mathbf{p}^2)$ involved in this constraint:

$$\{\mathbf{x} \in \{0,1\}^n : x_j = 0\ \forall j \notin N_i \cap D(\mathbf{p}^1, \mathbf{p}^2)\}.$$

Let $\mathbf{x}^{ik}, k = 1, \ldots, 2^{|N_i \cap D(\mathbf{p}^1, \mathbf{p}^2)|}$ denote the k-th vector in this set. For each combination k which violates a constraint i from (2), i.e.

$$\sum_{j \in N_i \cap D(\mathbf{p}^1, \mathbf{p}^2)} a_{ij} x_j^{ik} + \sum_{j \in N_i \setminus D(\mathbf{p}^1, \mathbf{p}^2)} a_{ij} p_j^1 > b_i,$$

we add an edge

$$e_{ik} = \{v_j : x_j^{ik} = 1,\ j \in N_i \cap D(\mathbf{p}^1, \mathbf{p}^2)\} \cup \{v_{j+n} : x_j^{ik} = 0,\ j \in N_i \cap D(\mathbf{p}^1, \mathbf{p}^2)\}$$

into the hypergraph. (Note that the edge e_{ik} contains at most $|N_i|$ elements.) Besides, we add d edges $\{v_j, v_{n+j}\}, j \in D(p_1, p_2)$ to guarantee that both v_j and v_{n+j} will not enter into an independent set together.

If $\mathbf{x}$ is a feasible solution to the ORP for (1)-(3) then, the set of vertices

$$S(\mathbf{x}) = \{v_j : x_j = 1, j \in D(p_1, p_2)\} \cup \{v_{j+n} : x_j = 0, j \in D(p_1, p_2)\}$$

is independent in H. Given a set of vertices S, we can construct the corresponding vector $\mathbf{x}(S)$, assigning $\mathbf{x}(S)_j = 1$ if $v_j \in S$, $j \in D(\mathbf{p}^1, \mathbf{p}^2)$ or if $p_j^1 = p_j^2 = 1$. Otherwise, $\mathbf{x}(S)_j = 0$. Then, for each independent set S of d vertices, $\mathbf{x}(S)$ is feasible in the Boolean Linear Programming Problem.

The hypergraph vertices are given the following weights:

$$w(v_j) = c_j + \lambda,\ w(v_{n+j}) = \lambda,\ j \in D(\mathbf{p}^1, \mathbf{p}^2),$$

where $\lambda > 2\sum_{j \in D(p_1, p_2)} |c_j|$.

Now each maximum weight independent set S^* contains either v_j or v_{n+j} for any $j \in D(\mathbf{p}^1, \mathbf{p}^2)$. Indeed, there must exist a feasible solution to the ORP and it corresponds to an independent set of weight, at least λd. However, if an independent set neither contains v_j nor v_{n+j} then, its weight is below $\lambda d - \lambda/2$.

So, optimal independent set S^* corresponds to a feasible vector $\mathbf{x}(S^*)$ with the goal function value

$$\mathbf{cx}(S^*) = \sum_{j \in S^*,\ j \le n} c_j + \sum_{j \notin D(\mathbf{p}^1, \mathbf{p}^2)} c_j p_j^1 = w(S^*) - \lambda d + \sum_{j \notin D(\mathbf{p}^1, \mathbf{p}^2)} c_j p_j^1.$$

Under the mapping $S(\mathbf{x})$, which is inverse to $\mathbf{x}(S)$, any feasible vector $\mathbf{x}$ yields an independent set of weight

$$w(S(\mathbf{x})) = \mathbf{cx} + \lambda d - \sum_{j \notin D(\mathbf{p}^1, \mathbf{p}^2)} c_j p_j^1,$$

therefore $\mathbf{x}(S^*)$ is an optimal solution to the ORP. □

Note that if an edge $e \in H$ consists of a single vertex, $e = \{v\}$ then, the vertex v can not enter into the independent sets. All such vertices should be excluded from the hypergraph H constructed in Theorem 3.6. Let us denote the resulting hypergraph by H'. If $N_{\max} \le 2$ then, the hypergraph H' is an ordinary graph with at most $2d$ vertices. Thus, by Theorem 3.6 the ORP reduces to the Maximum Weight Independent Set Problem in a bipartite graph H', which is solvable in $O(d^3)$ operations. Using this fact, Theorem 3.1 may be extended as follows:

Corollary 3.7 [15] *The ORP for Linear Boolean Programming Problem with at most two variables per inequality is solvable in time* $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3 + mn)$ *if the solutions are represented by vectors* $x \in \{0, 1\}^n$.

The class of Linear Boolean Programming Problems with at most two variables per inequality includes the Vertex Cover Problem and the *Minimum 2-Satisfiability Problem* – see e.g [19].

4 NP-HARD OPTIMAL RECOMBINATION PROBLEMS IN BOOLEAN LINEAR PROGRAMMING

It was shown above that the optimal recombination on the class of Boolean Linear Programming Problems is related to the Maximum Weight Independent Set Problem on hypergraphs with a given 2-coloring. The next lemma indicates that in general case the latter problem is NP-hard.

Lemma 4.1 [15] *The problem of finding a maximum size independent set in a hypergraph with all edges of size 3 is strongly NP-hard even if a 2-coloring is given.*

Proof. Let us construct a reduction from the strongly NP-hard Maximum Size Independent Set Problem on ordinary graphs to the problem under consideration. Given a graph $G = (V, E)$ with the set of vertices $V = \{v_1, \ldots, v_n\}$, consider a hypergraph $H = (V', E')$ on the set of vertices $V' = \{v_1, \ldots, v_{2n}\}$, where for each edge $e = \{v_i, v_j\} \in E$ there are n edges of the form $\{v_i, v_j, v_{n+k}\},\ k = 1, \ldots, n$ in E'. A 2-coloring for this hypergraph can be composed of color classes $C_1 = V$ and $C_2 = \{v_{n+1}, \ldots, v_{2n}\}$. Any maximum size independent set in this hypergraph consists of a set of vertices $\{v_{n+1}, \ldots, v_{2n}\}$ joined with a maximum size independent set S^* in G. Therefore, any maximum size independent set in H immediately induces a maximum size independent set for G. □

The Maximum Size Independent Set Problem in a hypergraph $H = (V, E)$ may be formulated as a Boolean Linear Programming Problem

$$\max \left\{ \sum_{j=1}^{n} x_j : \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \{0,1\}^n \right\} \tag{20}$$

with $m = |E|, n = |V|$, $b_i = |e_i| - 1,\ i = 1, \ldots, m$ and $a_{ij} = 1$ iff $v_j \in e_i$, otherwise $a_{ij} = 0$. In the special case where H is 2-colorable, we can take $\mathbf{p}^1$ and $\mathbf{p}^2$ as the indicator vectors for the color classes C_1 and C_2 of any 2-coloring. Then, $D(\mathbf{p}^1, \mathbf{p}^2) = \{1, \ldots, n\}$, and the ORP for the Boolean Linear Programming Problem (20) becomes equivalent to solving the maximum size independent set in a hypergraph H with a given 2-coloring. In view of Lemma 4.1, this leads to the following

Theorem 4.2 [15] *The optimal recombination problem for Boolean Linear Programming Problem is strongly NP-hard even in the case where $|N_i| = 3$ for all $i = 1, \ldots, m$; $c_j = 1$ for all $j = 1, \ldots, n$ and matrix $\mathbf{A}$ is Boolean.*

In the rest of this section, we will discuss NP-hardness of the ORPs for some well-known Boolean Linear Programming Problems.

4.1 One-Dimensional Knapsack and Bin Packing

In Boolean linear programming formulation the One-Dimensional Knapsack Problem has the following formulation

$$\max \{\mathbf{c}\mathbf{x} : \mathbf{a}\mathbf{x} \leq A, \mathbf{x} \in \{0,1\}^n\}, \tag{21}$$

where $\mathbf{c} = (c_1, \ldots, c_n)$, $\mathbf{a} = (a_1, \ldots, a_n)$, $a_j \geq 0, c_j \geq 0, j = 1, \ldots, n$, and $A \geq 0$ are integer.

Below, we also consider the *One-Dimensional Bin Packing Problem*. Given an integer number A (size of a bin) and k integer numbers $a_1, \ldots, a_k$ (sizes of items), $a_i \leq A,\ i = 1, \ldots, k$ it is required to locate the items into the minimal number of bins, so that the sum of sizes of items in each bin does not exceed A.

The One-Dimensional Bin Packing Problem may be formulated as a Boolean Linear Programming Problem the following way (a more "standard" integer linear programming formulation can be found e.g. in [22]). Let a Boolean variable y_j be the indicator of usage of a bin j, $j = 1, \ldots, k$ and a Boolean variable x_{ij}, $i, j = 1, \ldots, k$ be the indicator of packing item i in bin j. Find

$$\min \sum_{j=1}^{k} y_j \tag{22}$$

s. t.

$$\sum_{j=1}^{k} x_{ij} = 1, \quad i = 1, \ldots, k, \tag{23}$$

$$\sum_{i=1}^{k} a_i x_{ij} \leq A, \quad j = 1, 2, \ldots, k, \tag{24}$$

$$y_j \geq x_{ij}, \quad i = 1, \ldots, k,\ j = 1, \ldots, k, \tag{25}$$

$$x_{ij},\ y_j \in \{0, 1\}, \quad i = 1, \ldots, k,\ j = 1, 2, \ldots, k. \tag{26}$$

Note that for solutions encoding, it suffices to store only the *matrix of assignments* (x_{ij}), since the vector $(y_1, \ldots, y_k)$ corresponding to such a matrix is uniquely defined. Below, we assume that $(k \times k)$-matrices of assignments are used to encode the feasible solutions and $n = k^2$.

The following special case of the well-known Partition Problem [17] will be called *Bounded Partition*: Given $2m$ positive integer numbers $\alpha_1, \ldots, \alpha_{2m}$, which satisfy

$$\frac{B}{m+1} < \alpha_j < \frac{B}{m-1}, \quad j = 1, \ldots, 2m, \tag{27}$$

where $B = \sum_{j=1}^{2m} \alpha_j / 2$, *is there* a vector $\mathbf{x} \in \{0, 1\}^{2m}$, such that

$$\sum_{j=1}^{2m} \alpha_j x_j = B? \tag{28}$$

The next lemma is due to P. Schuurman and G. Woeginger.

Lemma 4.3 [30] *The Bounded Partition Problem is NP-complete.*

Proof. NP-completeness of this problem may be established via reduction from the following NP-complete modification of Partition Problem [17]: given a set of $2m$ positive integers $\alpha'_j,\ j = 1, \ldots, 2m$, it is required to recognize existence of such $\mathbf{x} \in \{0,1\}^{2m}$, that

$$\sum_{j=1}^{2m} x_j = m \quad \text{and} \quad \sum_{j=1}^{2m} \alpha'_j x_j = \frac{1}{2} \sum_{j=1}^{2m} \alpha'_j. \tag{29}$$

The reduction consists in setting $\alpha_i = \alpha'_i + M,\ i = 1, \ldots, 2m$, with a sufficiently large integer M, e.g., $M = 2m \cdot \max\{\alpha'_j : j = 1, \ldots, 2m\}$. Satisfaction of (27), as well as equivalence of (29) and (28), given this set of parameters $\{\alpha_i\}$, is verified straightforwardly. $\square$

Theorem 4.4 [12] *The ORPs for the One-Dimensional Knapsack Problem (21) and the One-Dimensional Bin-Packing Problem (22) – (26) are NP-hard.*

Proof. 1. Consider ORP for Knapsack Problem (21). The NP-hardness of this problem can be established using a polynomial-time Turing reduction of Bounded Partition Problem to it. W. l. o. g. let us assume $m > 2$.

Note that if an instance of Bounded Partition Problem has the answer "yes", then there exists a vector $\mathbf{x}' \in \{0,1\}^{2m}$, such that $\sum_{i=1}^{2m} \alpha_i x'_i = B$, and since $B/(m+1) < \alpha_i < B/(m-1),\ i = 1, \ldots, 2m$, this vector contains exactly m ones, which is less than $2m-2$ because $m > 2$. On the contrary, if the instance of Partition Problem has the answer "no" then, such a vector does not exist.

The Turing reduction of Bounded Partition Problem to the ORP for One-Dimensional Knapsack problem is based on enumeration of polynomial number of different pairs of parent solutions (and the corresponding ORP instances). Assume $n = 2m$, $A = B$ and $c_j = a_j = \alpha_j,\ j = 1, \ldots, n$, and enumerate all of the $\binom{2m}{2}$ pairs of variables with indices $\{i_\ell, j_\ell\},\ \ell = 1, \ldots, \binom{2m}{2}$. For each pair i_ℓ, j_ℓ we set $p^1_{i_\ell} = p^2_{j_\ell} = 0$ and fill the remaining positions $j \notin \{i_\ell, j_\ell\}$ so that $p^1_j + p^2_j = 1$ and each of the parent solutions contains in total $m-1$ ones (such parent solutions will be feasible since $a_j < A/(m-1),\ j = 1, \ldots, n$). The greatest value among the optima of the constructed ORPs equals A iff the answer to the instance of Partition Problem is "yes". This implies NP-hardness of the ORP for One-Dimensional Knapsack Problem.

2. The proof of NP-hardness of the ORP for One-Dimensional Bin-Packing Problem is based on a similar (but more time demanding) Turing reduction from Bounded Partition Problem. Now we assume $k = 2m$, $A = B$, and $a_i = \alpha_i,\ i = 1, \ldots, k$. In what follows it is supposed that $m > 4$.

Given an instance of Bounded Partition Problem, we enumerate a polynomial number of parent solutions, choosing them in such a way that (i) $2m-4$ items in the offspring solution are packed into the first two containers, (ii) among them, a pair of "selected" items may be packed only in bin 2, (iii) four other "selected" items may be packed either in bin 1 or in bin 3, optionally. Let us describe this reduction in detail.

As in the first part of the proof, we enumerate all of the $\binom{2m}{2}$ pairs of items $\{i_\ell, i'_\ell\},\ \ell = 1, \ldots, \binom{2m}{2}$, aiming to fix the corresponding variables $\{x_{i_\ell,1}, x_{i'_\ell,1}\}$ to zero value.

For each of the pairs $\{i_\ell, i'_\ell\}$ enumerate all $\binom{2m-2}{2}$ pairs $\{u_r, u'_r\},\ r = 1, \ldots, \binom{2m-2}{2}$ drawn from the rest of the items. Given $\{i_\ell, i'_\ell\}$ and $\{u_r, u'_r\}$, enumerate all $\binom{2m-4}{2}$ pairs $\{v_s, v'_s\},\ s = 1, \ldots, \binom{2m-4}{2}$, in the rest of the items.

To ensure that for given ℓ, r and s, the items $\{i_\ell, i'_\ell\}$ in the offspring solution are packed in bin 2, while items u_r, u'_r, v_s, v'_s may be packed only in bin 1 or bin 3, the pair of parent solutions $\mathbf{p}^1 = (p^1_{ij})$ and $\mathbf{p}^2 = (p^2_{ij})$ is defined the following way.

In the first column of parent solutions

$$p^1_{i_\ell,1} = p^1_{i'_\ell,1} = p^2_{i_\ell,1} = p^2_{i'_\ell,1} = 0,$$

$$p^1_{u_r,1} = p^1_{u'_r,1} = 1,\ p^2_{u_r,1} = p^2_{u'_r,1} = 0,$$

$$p^1_{v_s,1} = p^1_{v'_s,1} = 0,\ p^2_{v_s,1} = p^2_{v'_s,1} = 1$$

and fill the remaining positions $i \notin \{i_\ell, i'_\ell, u_r, u'_r, v_s, v'_s\}$ so that $p^1_{i,1} + p^2_{i,1} = 1$ holds and each of the parent solutions has $m-1$ ones in column 1. These parent solutions satisfy condition (24) for bin $j = 1$, since $a_i < A/(m-1)$, $i = 1, \ldots, k$.

Let the second column in each of the parent solutions be identical to the first column of the other parent, except for the components corresponding to the six items mentioned above. Two entries 1 in rows v_s and v'_s of the parent solution $\mathbf{p}^1$ are placed into column $j = 3$, rather than into column $j = 2$. Two entries 1 in rows u_r and u'_r of the parent solution $\mathbf{p}^2$ are placed into column $j = 3$, rather than into column $j = 2$. Besides, in column $j = 2$ of both parent solutions the entries 1 are placed in rows i_ℓ and i'_ℓ.

In each parent solution, the second column contains $m-1$ entries 1 thus, condition (24) for bin $j = 2$ is satisfied, as well as in the case of $j = 1$. For bin $j = 3$, this condition holds since $a_i < A/4$, $i = 1, \ldots, k$ when $m > 4$. Note that all feasible solutions to the ORP corresponding to a triple of indices ℓ, r, s contain the items i_ℓ, i'_ℓ in the second bin, while items u_r, u'_r, v_s and v'_s may appear either in bin 1 or in bin 3.

If an instance of Bounded Partition Problem has the answer "yes" then, at least one of the constructed ORPs has the optimum objective function value 2. Indeed, in such a case the vector $\mathbf{x}'$ that satisfies condition (28) should have two entries $x'_{\hat{i}} = x'_{\bar{i}} = 0$ for some $\hat{i}, \bar{i}$. Besides, there are four indices $\hat{u}, \bar{u}, \hat{v}$ and $\bar{v}$ such that $x'_{\hat{u}} = x'_{\bar{u}} = x'_{\hat{v}} = x'_{\bar{v}} = 1$, since this vector contains not less than m entries 1. The corresponding ORP with $\{i_\ell, i'_\ell\} = \{\hat{i}, \bar{i}\}$, $\{u_r, u'_r\} = \{\hat{u}, \bar{u}\}$ and $\{v_s, v'_s\} = \{\hat{v}, \bar{v}\}$ has a feasible solution (x'_{ij}), where the first column is identical to $\mathbf{x}'$, the entries of the second column are $x'_{i2} = 1 - x'_i,\ i = 1, \ldots, k$, and the rest of the columns are filled with zeros.

Conversely, if an optimal solution x^*_{ij} to one of the constructed ORPs has value 2 then, setting $x_i = x^*_{i1},\ i = 1, \ldots, k,$, we obtain equality (28). □

The One-Dimensional Bin Packing problem is contained, as a special case, in a number of packing and scheduling problems so, the latter theorem may be applicable in analysis of complexity of the ORPs for these problems. In particular, Theorem 4.4 implies NP-hardness of the ORP for the Transfer Line Balancing Problem [13].

4.2 Set Covering and Location Problems

The next example of an NP-hard ORP is that for the Set Covering Problem, which may be considered as a special case of (1)-(3):

$$\min\{\mathbf{cx} : \mathbf{Ax} \geq \mathbf{e},\ \mathbf{x} \in \{0,1\}^n\}, \tag{30}$$

where $\mathbf{A}$ is a Boolean $(m \times n)$-matrix; $\mathbf{c} = (c_1, \ldots, c_n)$; $c_j \geq 0,\ j = 1, \ldots, n$. Let us assume that the binary representation of solutions by the vector $\mathbf{x}$. Given an instance of the Set Covering Problem, one may construct a new instance with a doubled set of columns in the matrix $\mathbf{A}' = (\mathbf{AA})$ and a doubled vector $\mathbf{c}' = (c_1, \ldots, c_n, c_1, \ldots, c_n)$. Then, an instance of the NP-hard Set Covering Problem (30) is equivalent to the ORP for the modified set covering instance where the input consists of $(m \times 2n)$-matrix $\mathbf{A}'$, $2n$-vector $\mathbf{c}'$ and the feasible parent solutions $\mathbf{p}^1, \mathbf{p}^2$, with $p^1_j = 1, p^2_j = 0$ for $j = 1, \ldots, n$ and $p^1_j = 0, p^2_j = 1$ for $j = n+1, \ldots, 2n$. So, the ORP for the Set Covering Problem is also NP-hard.

Interestingly, in some cases the ORP may be even harder than the original problem (assuming P$\neq$ NP). This can be illustrated on the example of the Set Covering Problem. A special case of this problem, defined by the restriction $a_{i,1} = 1, i = 1, \ldots, m;\ c_1 = 0$ is trivially solvable: $\mathbf{x} = (1, 0, 0, \ldots, 0)$ is the optimal solution. However, in the case $p^1_1 = p^2_1 = 0$, the ORP becomes NP-hard under this restriction.

The Set Covering Problem may be efficiently transformed to the Simple Plant Location Problem (10)-(11) – see e.g. transformation T3 in [24]. In this case the dimensions m and n in both problems are equal, $C_i = c_i$ for $i = 1, \ldots, n$ and

$$c_{ij} = \begin{cases} \sum_{k=1}^{n} c_k + 1, & \text{if } a_{ij} = 0, \\ 0, & \text{if } a_{ij} = 1, \end{cases} \quad \text{for all } i = 1, \ldots, n,\ j = 1, \ldots, m.$$

It is easy to verify that a vector $\mathbf{x}^*$ in the optimal solution to this instance of the Simple Plant Location Problem will be an optimal set covering solution as well. Thus, if the solution representation in the Simple Plant Location Problem is given only by the vector $\mathbf{x}$ then, this reduction meets the conditions of Proposition 3.3. The subset of solutions to the Simple Plant Location Problem $\beta(\mathrm{Sol}_1(I))$ is characterized in this case by the threshold on objective function $f_{\mathrm{splp}}(\mathbf{y}) < \sum_{j=1}^{n} c_j + 1$, which ensures that all constraints of the Set Covering Problem are met. Therefore, an NP-hard ORP problem is efficiently reduced to the ORP for (10)-(11) and the following proposition holds.

Proposition 4.5 [15] *The ORP for the Simple Plant Location Problem (10), (11) is NP-hard.*

The well-known p-Median Problem may be defined as a modification of the Simple Plant Location Problem (10), (11): it suffices to assume $C_k = 0,\ j = 1, \ldots, n$, and to substitute the inequality (11) by constraint

$$\sum_{i=1}^{n} x_i = p, \tag{31}$$

where $1 \leq p \leq n$ is a parameter given in the problem input.

Proposition 4.6 [15] *The ORP for the p-Median Problem (10), (31) is NP-hard.*

Proof. E. Alexeeva, Yu. Kochetov and A. Plyasunov in [2] propose a reduction of an NP-hard Graph Partitioning Problem to the p-Median Problem with $n = |V|$ and $p = |V|/2$, where V is the set of the graph vertices and $|V|$ is even. Thus, this special case of the p-Median Problem is NP-hard as well. Consider an ORP for this case of the p-Median Problem with parent solutions $\mathbf{p}^1 = (1, \ldots, 1, 0, \ldots, 0)$ and $\mathbf{p}^2 = (0, \ldots, 0, 1, \ldots, 1)$ of $n/2$ ones. Obviously, such ORP is equivalent to the original p-Median Problem. □

REFERENCES

[1] Agarwal, C.C., Orlin, J.B. and Tai, R.P., *Optimized crossover for the independent set problem*, Massachusetts Institute of Technology, 1995.

[2] Alekseeva, E., Kochetov, Yu. and Plyasunov, A., "Complexity of local search for the p-median problem", *European Journal of Operational Research*, 191 (2008) 736-752.

[3] Ausiello, G., Crescenzi, P., Gambosi, G. et al., *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, Springer-Verlag,Berlin, 1999.

[4] Balas, E. and Niehaus, W., *A Max-Flow Based Procedure for Finding Heavy Cliques in Vertex-Weighted Graphs*, MSRR no. 612, Carnegie-Mellon University, 1995.

[5] 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.

[6] Beresnev, V.L., Gimady, Ed.Kh. and Dementev, V. T., *Extremal Problems of Standardization*, Novosibirsk, Nauka, 1978 (in Russian).

[7] Beyer, H.-G., Schwefel, H.-P. and Wegener, I., "How to analyse evolutionary algorithms", *Theoretical Computer Science*, 287 (2002) 101-130.

[8] Borisovsky, P., Dolgui, A. and Eremeev, A., "Genetic algorithms for a supply management problem: MIP-recombination vs greedy decoder", *European Journal of Operational Research*, 195 (3) (2009) 770-779.

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

[10] Cotta, C., "A study on allelic recombination", *Proc. of 2003 Congress on Evolutionary Computation*, Canberra, IEEE Press, 2003, 1406-1413.

[11] Cotta, C. and Troya, J.M., "Embedding branch and bound within evolutionary algorithms", *Applied Intelligence* 18 (2003) 137-153.

[12] Dolgui, A. and Eremeev, A., "On complexity of optimal recombination for one-dimensional bin packing problem", *Proc. of VIII Intern. Conf. "Dynamics of Systems, Mechanisms and Machines"*, Vol. 3, Omsk, Omsk Polytechnical University, 2012, 25-27. (In Russian)

[13] 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.

[14] Eremeev, A.V., "A Genetic algorithm with a non-binary representation for the set covering problem", *Proc. of Operations Research (OR'98)*, Springer-Verlag, Berlin,1999, 175-181.

[15] Eremeev, A.V., "On complexity of optimal recombination for binary representations of solutions", *Evolutionary Computation*, 16 (1) (2008) 127-147.

[16] 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)

[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] Glover, F., Laguna, M. and Marti, R., "Fundamentals of scatter search and path relinking", *Control and Cybernetics*, 29 (3) (2000) 653-684.

[19] Hochbaum, D.S. "Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems", *Approximation Algorithms for NP-Hard Problems,* (ed.) by D. Hochbaum, PWS Publishing Company, Boston,1997, 94-143.

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

[21] Jansen, T. and Wegener, I., "On the analysis of evolutionary algorithms – a proof that crossover really can help", *Algorithmica*, 34 (1) (2002) 47-66.

[22] Mukhacheva, E.A. and Mukhacheva, A.S., "L.V. Kantorovich and cutting-packing problems: New approaches to combinatorial problems of linear cutting and rectangular packing", *Journal of Mathematical Sciences*, 133 (4) (2006) 1504-1512.

[23] Korte, B. and Vygen, J., *Combinatorial Optimization. Theory and Algorithms*, 3rd edition, Springer-Verlag, Berlin,2005.

[24] Krarup, J. and Pruzan, P., "The simple plant location problem: survey and synthesis", *European Journal of Operational Research*, 12 (1983) 36-81.

[25] Lourenγ*co*, *H.*, *Paix*ão, J. and Portugal, R., "Multiobjective metaheuristics for the bus driver scheduling problem", *Transportation Science,* 35 (2001) 331-343.

[26] Papadimitriou, C.H. and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Upper Saddle River, NJ,1982.

[27] Radcliffe, N.J., "The algebra of genetic algorithms", *Annals of Mathemathics and Artificial Intelligence*, 10 (4) (1994) 339-384.

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

[29] Reeves, C.R. and Rowe, J.E., *Genetic algorithms: principles and perspectives,* Norwell, MA, Kluwer Acad. Pbs., 2002.

[30] Schuurman, P. and Woeginger, G., *Approximation schemes – a tutorial*, Eindhoven University of Technology, 2006.

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

[32] Yankovskaya, A.E., "Test pattern recognition with the use of genetic algorithms", *Pattern Recognition and Image Analysis*, 9 (1) (1999) 121-123.