# Genetic Algorithms for a Supply Management Problem: MIP-Recombination vs Greedy Decoder

P. Borisovsky

*Omsk State Technical University, Prospect Mira 11, 644050 Omsk, Russia.*

A. Dolgui

*Ecole des Mines de Saint Etienne, 158, Cours Fauriel 42023 Saint Etienne cedex 2, France.*

A. Eremeev *

*Omsk Branch of Sobolev Institute of Mathematics, Siberian Branch of Russian Academy of Sciences, Pevtsov St., 13, 644099, Omsk, Russia*

**Abstract**

Two variants of genetic algorithm (GA) for solving the Supply Management Problem with Lower-Bounded Demands (SMPLD) are proposed and experimentally tested. The SMPLD problem consists in planning the shipments from a set of providers to a set of consumers minimizing the total cost, given lower and upper bounds on shipment sizes, lower-bounded consumption and linear costs for opened deliveries. The first variant of GA uses the standard binary representation of solutions and a new recombination operator based on the mixed integer programming (MIP) techniques. The second GA is based on the permutation representation and a greedy decoder. Our experiments indicate that the GA with MIP-recombination compares favorably to the other GA and to the MIP-solver CPLEX 9.0 in terms of cost of obtained solutions. The GA based on greedy decoder is shown to be the most robust in finding feasible solutions.

*Key words:* Genetic Algorithms, Mixed Integer Programming, Recombination

* Corresponding author.

*Email addresses:* borisovski@mail.ru (P. Borisovsky), dolgui@emse.fr (A. Dolgui), eremeev@iitam.omsk.net.ru (after 01.03.2007: eremeev@ofim.oscsbras.ru) (A. Eremeev).

## 1 Introduction

This paper presents a new variant of genetic algorithm for solving the Supply Management Problem with Lower-Bounded Demands (SMPLD) initially formulated in (Chauhan *et al.* (2004)). In this problem a set of providers supply product of one type to a set of consumers, the quantity of product that can be delivered lies between the given minimum and maximum values, and the costs proposed by each provider are linear functions of the quantity being delivered if the shipment is opened. The SMPLD consists in finding the quantity of product which will be supplied by each provider to each consumer minimizing the total cost

The mathematical formulation of SMPLD is:

$$\text{Min} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} z_{ij}(a_{ij} + c_{ij}x_{ij}), \tag{1}$$

$$\sum_{i=1}^{m} x_{ij} \geq A_j, \ j = 1, ..., n, \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} \leq M_i, \ i = 1, ..., m, \tag{3}$$

$$z_{ij} \in \{0, 1\}, \ i = 1, ..., m, \ j = 1, ..., n, \tag{4}$$

$$m_{ij}z_{ij} \leq x_{ij} \leq M_i z_{ij}, \ i = 1, ..., m, \ j = 1, ..., n. \tag{5}$$

Here $m$ is the number of suppliers; $n$ is the number of consumers; variables $z_{ij}$ indicate the presence of delivery from a provider $i$ to a consumer $j$, while $x_{ij}$ represent the shipment size. $A_j$ is the minimal amount of product required for the consumer $j$; $m_{ij}$ is the minimum quantity the provider $i$ is prepared to deliver to the consumer $j$; $M_i$ is the maximum quantity the provider $i$ is able to deliver in total. Parameters $a_{ij}, c_{ij}, m_{ij}, M_i, A_j$ are supposed to be non-negative and integer, although the integrality is not essential in this paper.

Many practical problems in logistics, scheduling and production planning can be modeled as SMPLD or contain it as an important special case – see e.g. (Sun *et al.* (1998); Sheng *et al.* (2006); Gaspero *et al.* (2006)). An alternative view on the SMPLD problem can be described as follows. Suppose there is a single consumer, which orders some perishable product for a planning horizon of $n$ days. For each day $j$ the consumption level $A_j$ is known and the excess of product delivered on day $j$ can not be stored for day $j + 1$, but it is disposed of. The assumptions concerning the optimization criterion and providers restrictions are the same as in the first problem formulation, except that now $M_i$ defines the maximal total delivery of provider $i$ over the planning horizon.

In case all $m_{ij} = 0$ and the overall supply is equal to the overall demand, i.e. $\sum_{i=1}^{m} M_i = \sum_{j=1}^{n} A_j$, this problem becomes the well-known Fixed Charge Transportation Problem (FCTP) - see e.g. (Barr *et al.* (1981); Sun *et al.* (1998)). There are a number of publications addressing exact and approximate solution of FCTP. The exact methods are mainly based on the branch-and-bound and cutting plane techniques. The interested reader can find more details on these approaches in (Barr *et al.* (1981); Glover and Sherali (2005); Ortega and Wolsey (2003)) and in the references provided there. The heuristic approaches developed for the FCTP include the tabu search methods (Sun *et al.* (1998); Cranic and Gendreau (2002)), genetic algorithms (Eckert and Gottlieb (2002); Sheng *et al.* (2006)) and other techniques (Steinberg (1977)), most of which employ the fact that the set of vertices of FCTP linear relaxation polyhedron contains the optimal solution. Unfortunately, this fact does not apply to the SMPLD and here the heuristics should be based on some other problem properties. In addition, note that the presence of minimum shipment sizes in SMPLD can create a complex feasibility problem, which is absent in the case of FCTP.

The SMPLD is also closely related to the supply management problem (SMP) considered by Chauhan and Proth (2003) and Chauhan *et al.* (2005*a*). The first difference between SMP and SMPLD is that instead of inequality (2), in the SMP holds an equality

$$\sum_{i=1}^{m} x_{ij} = A_j, \;\; j = 1, ..., n. \tag{6}$$

The second difference is that in the SMP the cost functions are more general, namely, non-decreasing and concave. Finding any feasible solution to the SMP is NP-hard even for $n = 1$, yet there exists a pseudopolynomial-time exact algorithm for it (Chauhan *et al.* (2005*a*)). Some of the useful properties of SMP can also be transferred to the SMPLD. For example, it can be shown that analogously to SMP, any feasible SMPLD problem with a single consumer has an optimal solution where $x_{i1} \in \{0, m_{i1}, M_{i1}\}$ for all $i$, except for at most one coordinate.

The SMPLD is also NP-hard (it contains the minimum knapsack problem as a special case) and finding a feasible solution satisfying (2)–(5) is NP-hard even in the case $n = 2$. However, in case $n = 1$, this problem admits a fully polynomial time approximation scheme (FPTAS) and there is a fast greedy 2-approximation algorithm for it (Chauhan *et al.* (2004)). An FPTAS for $n = 1$ and more general form of the cost functions is developed in (Chauhan *et al.* (2005*b*)). At the same time, in the SMPLD, unlike SMP, consumption of goods is not given in advance, but only bounded from below, thus yielding larger linear relaxation polyhedron. So, for the branch-and-bound methods,

cutting planes, $L$-class enumeration (Kolokolov (1996)) and other algorithms based on the linear relaxation polyhedron, the SMPLD with inequality (2) should be practically more difficult, than its counterpart with equality (6).

The first genetic algorithm proposed in this paper (GAmipr) uses a natural binary representation of solutions and a new recombination operator, based on the mixed integer programming (MIP) techniques. This recombination operator aims at finding the best possible "offspring" of two given "parent" genotypes. The similar goals were achieved in the optimized crossover due to Aggarwal *et al.* (1997) and Balas and Niehaus (1998) for the maximum clique and maximum independent set problems by means of minimum cut or maximum matching algorithms. Incorporation of dynamic programming into crossover for optimal recombination also turned out to be successful in three problems on permutations: the single machine scheduling problem, the optimal linear arrangement and the traveling salesman problem (Yagiura and Ibaraki (1996)). Analogous goals of finding a near-optimal combination of solutions were addressed by means of path-relinking strategy in the framework of local search (Glover *et al.* (2000); Höhn and Reeves (1996)), which gave encouraging results for graph partitioning and other problems. In general, it appears that the approach based on optimal recombination is widely applicable. In the present paper we intend to solve the optimal recombination problem, relying on a modern MIP-solver, rather than developing the special-purpose recombination operators based on the problem-specific features. The flexibility of general-purpose MIP solvers and little use of the SMPLD features in the optimal recombination operator allow us to expect that this approach may be easily extended to a number of other problems.

The second GA suggested in this paper (GAgrd) is based on a greedy heuristic decoder and uses the permutation-based representation of solutions. The decoder applies the Greedy algorithm from (Chauhan *et al.* (2004)) separately to a subproblem of each consumer, ordering the consumers according to the permutation, encoded in the genotype string. A similar genetic algorithm for the SMP was developed and tested in Borisovsky (2004).

The outline of the paper is as follows. In Section 2 we describe the scheme of genetic algorithms being used in this paper, the binary representation of solutions of GAmipr, its fitness function and MIP-recombination operator. Also, this section contains an outline of the GAgrd and the greedy solutions decoder used in it. The testing instances and computational results are described in Section 3. Conclusions are given in Section 4. The present paper is an extended version of (Borisovsky *et al.* (2006)).

## 2 Genetic Algorithms

Genetic algorithm (GA) is a random search method that models a process of evolving a population of individuals, see e.g. (Dréo *et al.* (2006); Reeves (1997)). Each individual corresponds to some solution of the problem (feasible or maybe infeasible) and it is characterized by the *fitness*, reflecting the goal function value and the satisfaction of problem constraints. The higher is the fitness value, the more chances are given for the individual to be selected as a parent. New individuals are built by means of *reproduction* operator that consists of *crossover* and *mutation* procedures. The crossover procedure $Cross$ produces the offspring from two parent individuals by combining and exchanging their elements. The mutation procedure $Mut$ adds small random changes to an individual.

Let vector $\mathbf{\Pi}^{(t)}$ of $N$ solutions denote the current population on step $t$ in the GA. $N$ is called the population size and remains constant during the execution of a GA. The formal scheme of the GA with steady state replacement is as follows:

**Steady state GA**
1. Generate the initial population $\mathbf{\Pi}^{(0)}$ at random and set $t := 1$.
2. While a termination condition is not satisfied, do:
   2.1 $\mathbf{\Pi}^{(t)} := \mathbf{\Pi}^{(t-1)}$.
   2.2 Selection: choose $\mathbf{p}_1, \mathbf{p}_2$ from $\mathbf{\Pi}^{(t)}$.
   2.4 Apply mutation:
   $\mathbf{p}'_1 := Mut(\mathbf{p}_1)$, $\mathbf{p}'_2 := Mut(\mathbf{p}_2)$.
   2.3 Produce $\mathbf{c}_1$ and $\mathbf{c}_2$ applying the crossover:
   $(\mathbf{c}_1, \mathbf{c}_2) := Cross(\mathbf{p}'_1, \mathbf{p}'_2)$.
   2.5 Choose the two worst individuals $\mathbf{q}_1, \mathbf{q}_2$
   in $\mathbf{\Pi}^{(t)}$ and replace them by $\mathbf{c}_1$, $\mathbf{c}_2$.
   2.5 Set $t := t + 1$.

In this paper, we use the steady state replacement scheme of the GA and the choice of each parent on Step 2.2 is done by the $s$-tournament selection: take $s$ individuals by random from $\mathbf{\Pi}^{(t)}$ and select the best one.

The encodings of solutions in the GA are usually called *genotypes*. One of the most essential issues in development of a GA is the choice of representation of solutions in genotypes. Two different representations are considered in this paper.

### 2.1 Genetic Algorithm with a Binary Representation

A solution of SMPLD may be encoded in the genotype as a Boolean matrix $(z_{ij})$. It is easy to see that when all values of $(z_{ij})$ are fixed, the best possible feasible values of $(x_{ij})$ can be found by solving an LP-problem. However, this problem may have no feasible solutions. In order to deal with such cases we relax the original constraints and add the penalty terms to the criterion:

$$\text{Min } C_0 + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + R\left(\sum_{j=1}^{n}\alpha_j + \sum_{i=1}^{m}\beta_i\right), \tag{7}$$

$$\sum_{i=1}^{m} x_{ij} \geq A_j - \alpha_j,\ \ j = 1, ..., n, \tag{8}$$

$$\sum_{j=1}^{n} x_{ij} \leq M_i + \beta_i,\ \ i = 1, ..., m, \tag{9}$$

$$m_{ij}z_{ij} \leq x_{ij} \leq M_i z_{ij},\ \ i = 1, ..., m,\ \ j = 1, ..., n. \tag{10}$$

$$\alpha_j \geq 0,\ \ j = 1, ..., n,\ \ \beta_i \geq 0,\ \ i = 1, ..., m. \tag{11}$$

Here $C_0 = \sum_{i=1}^{m}\sum_{j=1}^{n} z_{ij}a_{ij}$ is the value of the fixed costs associated with genotype $\mathbf{z}$. Given a sufficiently large penalty parameter $R$, the term $\sum_{j=1}^{n}\alpha_j + \sum_{i=1}^{m}\beta_i$ in the optimal solution to problem (7)–(11) will be equal to 0 iff for the genotype $\mathbf{z}$ can be found some $(x_{ij})$, feasible with respect to problem (1)–(5). Let us assume the fitness function $\phi(\mathbf{z})$ to be the inverse of the optimal objective function value (7) in problem (7)–(11) with genotype $\mathbf{z}$.

Usually, the first choice for the genetic operators to use with binary representation is the one-point or uniform crossover and standard or one-bit-flip mutation (Reeves (1997)). Our preliminary experience with the SMP problem (Borisovsky (2004)) shows that the GAs based on the crossover and mutation of such type are rather inefficient for this problem due to relatively high computational cost of each fitness evaluation. To overcome this, we have developed a reproduction operator, called *MIP-recombination*, combining both mutation and crossover, which implicitly explores a large number of genotypes at a time by means of the modern integer programming techniques.

### 2.2 MIP-Recombination Operator

The first goal of the MIP-recombination operator is to find the best possible combination of two given parent genotypes $\mathbf{y}^1$ and $\mathbf{y}^2$, if it is possible without

extensive computations. The second goal is to add some random changes to the parent solutions, like in the traditional mutation procedures. In order to achieve both goals, we formulate a supplementary MIP problem, obtained from (1)–(5) by fixing all of the variables which have zero values in both parent genotypes, except for a random subset $S_{\text{mut}}$ of such variables:

$$z_{ij} = 0 \;\text{ for all } i,j: \; y^1_{ij} = y^2_{ij} = 0 \text{ and } (i,j) \notin S_{\text{mut}}. \tag{12}$$

Any pair $(i,j) : y^1_{ij} = y^2_{ij} = 0$ is independently included into $S_{\text{mut}}$ with probability $p_{mut}$. If an optimal solution to this problem $\mathbf{z}^*_{mipr}$ is found, then it becomes the output of MIP-recombination. In our implementation the MIP-solver of CPLEX 9.0 is used to find the optimum of subproblem (1)–(5), (12). To avoid time-consuming computations we set up a time limit $T_{rec}$ for each call to the solver. If the solver does not return a feasible solution then the MIP-recombination outputs a genotype $\text{Mut}(\mathbf{y}^1)$ obtained by the standard mutation procedure. Unlike the standard GA crossover, the described MIP-recombination procedure produces only one new individual at each iteration.

In what follows, the GA with the Boolean representation, fitness function $\phi(\mathbf{z})$ and MIP-recombination is called GAmipr. The genotypes of initial population in this GA are randomly generated by the same method as in the GAgrd described below. Another option we tested was to choose the components of the Boolean genotypes independently, assigning the value 1 with some probability, but this method turned out to be inefficient due to high probability of infeasible solutions.

In order to ensure the population diversity, we check whether the genotype $\mathbf{z}^*_{mipr}$ obtained in the MIP-recombination is already present in the current population $\mathbf{\Pi}^{(t)}$. If this is not the case, then $\mathbf{z}^*_{mipr}$ replaces an individual $q$ of lowest fitness in $\mathbf{\Pi}^{(t)}$. Otherwise, we substitute the genotype $q$ by $\text{Mut}(\mathbf{y}^1)$.

The initial value of mutation parameter $p_{mut}$ is set to $p^0_{mut}$ and adapted in the process of GA execution. Every time the MIP-recombination subproblem is solved to optimality, the parameter $p_{mut}$ is multiplied by 1.1. Whenever the solver is unable to find an optimal solution within the time limit $T_{rec}$, and $p_{mut} > p^0_{mut}$, parameter $p_{mut}$ is divided by 2. This adaptive mechanism keeps the complexity of the MIP-recombination problems close to the limit where the solver is able to obtain exact solutions within the given time $T_{rec}$.

### *2.3 Genetic Algorithm with a Non-Binary Representation*

The non-binary representation is based on a heuristic decoder which requires consecutive solving the problem, separately for each consumer. One way to

look at the reconstruction of solution from this encoding is to imagine a queue in which the consumers receive their deliveries, satisfying constraint (2), if it is possible, and minimizing the shipment cost. The genotype in this case is a permutation $\pi = (j_1, ..., j_n)$ of consumers in the queue.

Similar to (7) – (11) consider the following relaxed SMPLD formulation for one consumer. Let us fix some $j$ from $\{1, 2, ..., n\}$ and denote for simplicity $A = A_j, c_i = c_{ij}, a_i = a_{ij}, m_i = m_{ij}, x_i = x_{ij}, z_i = z_{ij}$. Instead of $M_i$ we will use notation $M'_i$ because some amount of product may be already delivered to other consumers. Then the relaxed one-consumer SMPLD is as follows:

$$\sum_{i=1}^{m}(a_i z_i + c_i x_i) + R\alpha + R\sum_{i=1}^{m}\beta_i \to \min, \tag{13}$$

$$\sum_{i=1}^{m} x_i \geq A - \alpha, \tag{14}$$

$$m_i z_i \leq x_i \leq M'_i z_i + \beta_i,\ i = 1, ..., m, \tag{15}$$

$$\alpha, \beta_i \in \mathbf{Z}^+,\ z_i \in \{0, 1\},\ x_i \in \mathbf{Z}^+,\ i = 1, ..., m. \tag{16}$$

Note that since all input parameters are integer, there exists an optimal solution with an integer vector $x$, whenever problem (1)–(5) is solvable. In what follows we will assume that all $x_{ij} \in \mathbf{Z}^+$, where $\mathbf{Z}^+$ is the set of non-negative integers. In this case we may choose sufficiently large $R$, so that the goal function value (13) of any feasible solution with $\alpha = 0, \beta_i = 0, i = 1, ..., m$ is less than the value of any solution with non-zero penalty terms.

Given some permutation $\pi = (j_1, ..., j_n)$, a solution $((x_{ij}), (z_{ij}))$ is built by the following decoding procedure:

**Decoder($\pi$)**
1. Assume $M'_i := M_i,\ i = 1, ..., m$.
2. For $k := 1$ to $n$ do:
   2.1 Solve the relaxed SMPLD (13)–(16) for consumer $j = j_k$.
   2.2 Reduce the capacities: $M'_i := M'_i - x_{ij_k},\ i = 1, ..., m$.
   2.3 Fix the obtained values for variables $x_{ij_k},\ i = 1, ..., m$.
3. Find the values $\{z_{ij}\}$ that suits to $\{x_{ij}\}$.

The mixed-integer programming problem (13)–(16) is NP-hard to solve to optimality, so in order to save the CPU time we apply a modification of a greedy 2-approximation algorithm (Chauhan *et al.* (2004)) to it. On each iteration of the Greedy algorithm we add the maximal possible amount of product from

the provider, selected by a greedy rule. This algorithm terminates with a feasible solution of zero penalty whenever such solution exists.

**Greedy Algorithm**
1. Assume $A' := A$ and $x_i := 0,\ i = 1, \ldots, m$.
2. While $x_i = 0,\ m_i \le M_i'$ for some $i$ and $A' > 0$ do:
   2.1 Let $S_i := \min\{\max\{A', m_i\}, M_i'\}$ for all $i = 1, \ldots, m$.
   2.2 Choose $i$ with the minimal value of $(a_i + c_i S_i)/\min\{A', M_i'\}$ among such $i$ that $x_i = 0$ and $m_i \le M_i'$.
   2.3 Set $x_i := S_i$ and $A' := A' - x_i$.

If after execution of the Greedy algorithm we have $A' < 0$ (which may occur on step 2.3 if $S_i > A'$), then the planned delivery exceeds the demand. An attempt to reduce this excess is made in the *refining procedure*. Here first we look for the small deliveries that can be closed without violating feasibility. If the excess remains then the size of some open deliveries is reduced where it is possible.

**Refining Procedure**
1. While $A' < 0$ and $m_i \le x_i \le -A'$ for some $i$ do:
   1.1 Choose $i$ such that $m_i \le x_i \le -A'$ and $a_i + c_i x_i$ is maximal.
   1.2 Set $A' := A' + x_i$ and $x_i := 0$.
2. While $A' < 0$ and $x_i > m_i$ for some $i$ do:
   1.1 Choose $i$ such that $x_i > m_i$ and $c_i$ is maximal.
   1.2 Set $d := \min\{x_i - m_i, -A'\}$, $x_i := x_i - d$, and $A' := A' + d$.

If $A' > 0$ after termination of the Greedy heuristic, then the solution has a non-zero penalty $RA'$. The described decoder can be used beyond the class of SMPLD problems (1)–(5) because the Greedy algorithm is applicable as a 2-approximation algorithm also in the case where the variable shipment costs are concave non-negative non-decreasing functions on $[0, M_i]$ (Chauhan *et al.* (2004)).

It is easy to see that not every feasible solution to the SMPLD problem can be represented by a permutation as described above. In fact, an example of solvable SMPLD instance can be constructed showing that even when the decoder solves all subproblems (13)–(16) to optimality, its output may be infeasible for any permutation $\pi$. Consider e.g. the $2 \times 2$ problem with

$$A_1 = A_2 = 3,\ M_1 = 4,\ M_2 = 2 \ \text{ and } \ (c_{ij}) = \begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix}, (m_{ij}) = \begin{pmatrix} 2 & 2 \\ 1 & 1 \end{pmatrix}.$$

The only feasible solution to the problem is $(x_{ij}) = (m_{ij})$. So if for the first consumer we optimally assign $x_{11} = 3, x_{21} = 0$, then the demand of the second consumer can not be satisfied. The same situation takes place if the Greedy starts from the second consumer.

To overcome (at least partially) the bias of the Greedy heuristic, a simple Local Improvement Heuristic has been proposed. Suppose four elements in matrix $(x_{ij})$ are chosen at random (at least two of them must be non-zero), then the corresponding subproblem of size $2 \times 2$ is easily solved exactly. If a solution is improved, then matrix $(x_{ij})$ is adjusted accordingly. In the Local Improvement Heuristic this random procedure is applied to each new individual 100 times, so this heuristic may be considered as a local search method with random examination of the neighborhood.

The non-binary representation described above is implemented in the genetic algorithm GAgrd. The initial population $\mathbf{\Pi}^{(0)}$ consists of $N$ randomly chosen permutations. This permutation encoding is similar to the encoding used in many GAs for the traveling salesman and scheduling problems. Thus the well-known *partially-mapped crossover* and *exchange mutation* are employed. The exchange mutation is performed as follows: given a permutation of consumers $\pi = (j_1, ..., j_n)$, choose two positions $k$ and $l$ by random and exchange $j_k$ and $j_l$. The exchange is made with probability $p_x$, otherwise $\pi$ remains unchanged after mutation. The partially-mapped crossover is performed with a fixed probability $p_{cross}$. This operator can be found e.g. in (Reeves (1997)).

## 3 Experimental Results

The GAgrd was programmed in Visual C++ 6.0 and the GAmipr was coded using OPL Script in ILOG OPL Studio 3.7. The experiments were carried out with the problem instances described below.

- **Series RAN** consists of 8 SMPLD instances obtained from FCTP benchmarks with given $m, n$ and $m_{ij} = 0$, constructed by Gottlieb and Paulmann (1998), using a modification of NETGEN generator (Barr *et al.* (1981)).
- **Series H** consists of 14 larger FCTP instances h_0, h_1, ... ,h_c and h_e (the instance h_d is omitted because it turned out to be identical to h_c) with $m = 100, n = 50$, $m_{ij} = 0$ and the range of fixed costs 6400-25600, produced by the same generator (Sun *et al.* (1998)).
- **Series S** consists of 8 randomly generated SMPLD instances with $m = n = 50$. All $m_{ij} = 10$, the values $M_i \in [400, 600]$, $A_j \in [350, 550]$, $a_{ij} \in [500, 4500]$ and $c_{ij} \in [20, 1020]$ are obtained by a uniformly distributed pseudo-random numbers generator. So, on average, the total supply is by 10% greater than the overall demand.

- **Series P** consists of 14 randomly generated SMPLD instances with $n = m = 50$. The values $M_i \in [12, 20]$, $A_j \in [10, 20]$, $a_{ij} \in [5, 10]$, $c_{ij} \in [10, 20]$, and $m_{ij} \in [5, 10]$ are obtained by a uniformly distributed generator. These ranges of parameters imply that on average the total supply is slightly higher than the overall demand. The values $m_{ij}$ are big enough to make it particularly difficult to find a feasible solution.
- **Series MIS** consists of 6 instances obtained from the maximum independent set problems, using the reduction proposed by Eremeev *et al.* (2006). In these SMPLD instances $m = |E| + 1$ and $n = |V|$, where $V$ and $E$ are the sets of vertices and edges in the graph of maximum independent set problem. The instances are obtained from the graphs of DIMACS collection of benchmarks (Johnson and Trick (1996)). The values of $m$, $n$ and the magnitude of coefficients $a_{ij}$ are given in Table 1.

FCTP benchmarks are available at *http://plato.la.asu.edu/ftp/fctp/*, and the graphs from DIMACS set can be found by
*ftp://dimacs.rutgers.edu/pub/challenge/graph/.*

In order to put all of the algorithms into equal positions, each of them was given the same amount of computation time $T$. Here and below the CPU time is given for 32-bit 3GHz Pentium-IV with 2.5 Gb RAM. For the smaller instances of series RAN and S we set $T = 2 \cdot 10^3$ seconds, while for the rest of the problems it was $10^4$ seconds. The GAs were restarted every time the termination condition halted them, until the overall execution time reached the limit $T$. The best solution found over all runs was returned as the result.

To compare the GAs to the modern branch-and-cut techniques, we ran CPLEX 9.0 solver built into ILOG OPL Studio 3.7 with the following modification of the default settings: `RINSHeur = 100, diveType = 3, MIPEmphasis = 1.` In what follows we will refer to these settings as *the heuristic mode.* The described settings activate the relaxation induced neighborhood search and guided dives (Danna *et al.* (2005)) and cause CPLEX emphasize searching for feasible solutions over proving optimality. This choice of parameters was the best option we found during our tests w.r.t. the quality of primal soultions. For comparison, the results of CPLEX 9.0 in the default mode can be found in (Borisovsky *et al.* (2006)).

In the experiments with the GAmipr we set the tournament size $s = 10$, the penalty factor $R = 10^{10}$ and the initial mutation probability $p^0_{mut} = 10^{-3}$. The time given for each call to the MIP-recombination operator is $T_{rec} = T/2000$, which implies that normally the GA will make several thousands of iterations in total. The CPLEX solver in this algorithm is called with the settings which differ from the heuristic mode only in parameter `MIPEmphasis = 0`, implying balance between feasibility and optimality. The population size was set to 500 for all problems, except for the instances of series MIS, where the OPL Studio

Table 1
Dimensions of the problems of series MIS

| Instance | $n$ | $m$ | $\max a_{ij}$ |
|---|---|---|---|
| hamming6.4 | 64 | 1313 | 4097 |
| c125.9 | 125 | 788 | 15626 |
| johnson16.2.4 | 120 | 1681 | 14401 |
| mann_a27 | 378 | 703 | 142885 |
| gen200_p0.9_44 | 200 | 1991 | 40001 |
| gen200_p0.9_55 | 200 | 1991 | 40001 |

memory limit did not allow to create a population of this size ($N = 100$ in mann_a27, gen200_p0.9_44 and gen200_p0.9_55; $N = 200$ in johnson16.2.4; $N = 400$ in c125.9 and hamming6.4).

Let $\theta$ be the time since the start of the current run of GAmipr, till the latest solution improvement. The termination condition of GAmipr is: restart if during the last $\theta$ seconds there was no best-found solution improvement and $t > N$.

The number of runs of GAmipr varied in a wide range. It turned out to be equal to 1 on all problems of series H. On series MIS it was between 2 and 4; from 17 to 25 on series S; from 101 to 329 on series RAN and from 1 to 71 on series P. The average number of iterations per run of GAmipr on series S and H was 614.2 and 3263.9 respectively. On series RAN it was ranging from 454.3 to 1720.0 iterations; on series MIS – between 258.7 (gen200_p0.9_44) and 435.0 (hamming6.4). On the problems of series P, when GAmipr found no feasible solutions, the number of iterations was very unstable (from 638.7 to 14209.0). For the rest of the problems of this series it was ranging between 987.50 and 3531.0. Such a difference in behavior on series P is due to variability of time required by CPLEX when the recombination problem is infeasible. Usually when the feasible solution is easy to obtain, more than 90% of CPU time in GAmipr is spent on solving the recombination problems.

In the GAgrd $s = 20$, $R = 10^7$, $p_x = 0.9$, $N = 10^3$, $p_{cross} = 0.8$ for all of the test problems. The termination condition in this GA is different from the one in GAmipx: on each problem the GAgrd was given equal time slots of $T/10$ seconds for 10 independent runs.

The best solutions found by the described algorithms are reported in Tables 2-6. Here $f^h_{cplex}$ corresponds to the results of stand-alone CPLEX solver with the heuristic settings described above. The best objective values obtained in our experiments are indicated in bold. The best-known solution values are given in

Table 2
Results for series RAN

| | $f_{best}$ | $f^h_{cplex}$ | $f_{GAmipr}$ | $f_{GAgrd}$ |
|---|---|---|---|---|
| ran13x13 | 3252* | **3252** | 3271 | 3284 |
| ran12x21 | 3664* | 3675 | **3664** | 3671 |
| ran14x18 | 3712* | 3746 | **3712** | 3714 |
| ran4x64 | 9711* | **9711** | **9711** | 9726 |
| ran8x32 | 5247* | **5247** | **5247** | **5247** |
| ran16x16 | 3823* | 3827 | **3823** | **3823** |
| ran10x26 | 4270* | **4270** | **4270** | **4270** |
| ran17x17 | 1373* | **1373** | **1373** | **1373** |

Table 3
Results for series S

| | $LB$ | $f_{best}$ | $f^h_{cplex}$ | $f_{GAmipr}$ | $f_{GAgrd}$ |
|---|---|---|---|---|---|
| s_1 | 604685 | 613951 | **613951** | **613951** | 614289 |
| s_2 | 622470 | 631219 | **631219** | **631219** | 631540 |
| s_3 | 617403 | 627294 | **627294** | **627294** | 627392 |
| s_4 | 622761 | 634609 | 636855 | **634609** | 634872 |
| s_5 | 595627 | 605145 | **605145** | 605380 | 605432 |
| s_6 | 655595 | 669346 | **669371** | 669787 | 670730 |
| s_7 | 589851 | 602737 | **602737** | **602737** | 603788 |
| s_8 | 633324 | 645843 | **645843** | **645843** | 647242 |

the column $f_{best}$ (the ones known as optimal are marked with "$*$"). Whenever an algorithm terminated without a feasible solution, we put "-". Tables 3 and 4 also contain the lower bounds (LB) found by CPLEX 9.0 run in the default mode for the same amount of time as the other algorithms. For the instances of series RAN, H and MIS, which are constructed using reductions, the values $f_{best}$ are obtained from the corresponding solutions of the FCTP and the maximum independent set problems.

The optimal solutions for series MIS are well known from the literature since they are easy to obtain by the problem-specific algorithms, solving the maximum independent set problem – see e.g. (Johnson and Trick (1996)).

For series H the best known solution costs are those obtained by the genetic algorithm GA NLO-1, especially developed for the FCTP by Eckert and Got-

Table 4
Results for series H

| | $LB$ | $f_{best}$ | $f^h_{cplex}$ | $f_{GAmipr}$ | $f_{GAgrd}$ |
|---|---|---|---|---|---|
| h_0 | 1002063 | 1208672 | 1260048 | **1217305** | 1284070 |
| h_1 | 990406 | 1204194 | 1241812 | **1231028** | 1303920 |
| h_2 | 991110 | 1192687 | 1269582 | **1198609** | 1305210 |
| h_3 | 954486 | 1188904 | 1236705 | **1215025** | 1305520 |
| h_4 | 992374 | 1209357 | 1283998 | **1238455** | 1332310 |
| h_5 | 997677 | 1202258 | 1264337 | **1214291** | 1311000 |
| h_6 | 990739 | 1201097 | 1258163 | **1235884** | 1305000 |
| h_7 | 967693 | 1172931 | 1259481 | **1204284** | 1281000 |
| h_8 | 984973 | 1198108 | 1272177 | **1236018** | 1306000 |
| h_9 | 972743 | 1192529 | 1267414 | **1227936** | 1308000 |
| h_a | 994451 | 1233688 | 1294036 | **1259553** | 1304000 |
| h_b | 1003666 | 1208211 | 1289744 | **1237080** | 1319000 |
| h_c | 990923 | 1209685 | 1260134 | **1233882** | 1338000 |
| h_e | 987160 | 1185378 | 1257240 | **1195558** | 1326000 |

tlieb (2002). On each instance of series H this algorithm was run 15 times until $10^7$ non-duplicate solutions were generated. (The run times of GA NLO-1 are unknown to us.)

The best known solution costs reported for series P are obtained by the algorithms described above or by the GAgrd with $N = 5 \cdot 10^4$ and similar restart rule as in GAmipr. The feasible solution to problem p8 was obtained only by a modification of GAgrd, followed by a call to CPLEX, that minimized a penalty-based objective function, starting from the GA solution.

While solving the instances p_3, gen200_p0.9_44 and gen200_p0.9_55 the standalone CPLEX solver ran out of memory and halted. The instance mann_a27 turned out to be unsolvable by CPLEX which returned an infeasible solution (probably, the error is caused by very large values of $a_{ij}$). The calls to CPLEX in GAmipr also frequently yielded infeasible solutions for mann_a27 and caused inadequate performance of the GA.

As a summary, Table 7 gives the *"gap"* values defined as $r = (f/f_{best} - 1) \cdot 100\%$, where $f$ is the goal function value found by the corresponding algorithm. The "gap" values are averaged over all problems in a series, except that instance mann_a27 is not counted in series MIS.

Table 5
Results for series P

| | $f_{best}$ | $f^h_{cplex}$ | $f_{GAmipr}$ | $f_{GAgrd}$ |
|---|---|---|---|---|
| p_1 | 8527 | - | **8527** | 8597 |
| p_2 | 7836 | **7836** | 7840 | 7865 |
| p_3 | 8423 | - | **8423** | 8536 |
| p_4 | 8409 | - | - | **8514** |
| p_5 | 8007 | 8011 | **8007** | 8027 |
| p_6 | 8320 | 8369 | **8320** | 8392 |
| p_7 | 7874 | **7874** | 7876 | 7889 |
| p_8 | 9543 | - | - | - |
| p_9 | 8831 | - | - | **8853** |
| p_10 | 8524 | - | **8524** | 8603 |
| p_11 | 8196 | - | **8196** | 8263 |
| p_12 | 8323 | - | **8323** | 8547 |
| p_13 | 8333 | - | 8696 | **8426** |
| p_14 | 8928 | - | - | **8928** |

Table 6
Results for series MIS

| | $f_{best}$ | $f^h_{cplex}$ | $f_{GAmipr}$ | $f_{GAgrd}$ |
|---|---|---|---|---|
| hamming6-4 | 60* | **60** | **60** | **60** |
| c125.9 | 91* | **91** | **91** | 94 |
| johnson16.2.4 | 112* | **112** | **112** | **112** |
| mann.a27 | 252* | - | 262 | **259** |
| gen200_p0.9_44 | 156* | 166 | **164** | 173 |
| gen200_p0.9_55 | 145* | **164** | 165 | 172 |

Tables 2–6 show that in general, on 50 instances the output of GAmipr was no worse than the result of CPLEX in 44 cases and in 28 among these cases the GAmipr performed better. The average relative error of GAmipr is less than that of the two other algorithms on all series (see Table 7).

The GAgrd turnes out to be less effective than the GAmipr and CPLEX on series RAN, S and H, where feasible solutions are relatively easy to find. On series P and MIS it shows the most robust performance, finding feasible

Table 7
Summary of average "gaps"

| Series | $r^h_{cplex}$ | $r_{GAmipr}$ | $r_{GAgrd}$ |
|---|---|---|---|
| RAN | 0.17 % | 0.07 % | 0.17 % |
| S | 0.05 % | 0.02 % | 0.10 % |
| H | 4.80 % | 2.00 % | 9.06 % |
| MIS | 9.76 % | 9.46 % | 15.85 % |

solutions in 19 out of 20 cases, while CPLEX failed in 11 cases and GAmipr failed 4 times. Still, in most of the problems where the other algorithms were able to find feasible solutions, the GAgrd was not the most precise one.

Figs 1-2 show the results for problem p_5. Fig. 1 displays the evolution of the cost of the best found solution vs the CPU time for each run and the dynamics of adaptively chosen value of $p_{mut}$. As it is seen from Fig. 1b, $p_{mut}$ increases on the initial stage of each run of the GAmipr. When the improvements of objective function no longer occur, $p_{mut}$ attains a stationary behavior.

Fig. 2 reports the best solutions from the beginning of the algorithms execution. It shows that GAmipr and CPLEX have similar curves on this problem and although GAgrd is the first one to find a feasible solution, it is quickly outperformed by the two other algorithms. Note that the tendency, analogous to that shown for problem p_5, was typical for series S and for other problems of series P, whenever all three of the algorithms were able to find feasible solutions.

In series H the GAmipr made no restarts and the behavior of $p_{mut}$ is quite different in Fig. 3b, compared to Fig. 1b. Here the mutation probability stays close to the level $p^0_{mut}$ during the first stage, where a high diversity of population is still inherited from the initial, randomly constructed population $\mathbf{\Pi}^{(0)}$. Subsequently both the mean level and the variance of $p_{mut}$ increase, and the mutation becomes the main source of population diversity. As it is seen from Fig. 3a, the GAmipr and CPLEX follow a similar trajectory in the beginning, but later the GAmipr becomes more successful. The GAgrd is clearly inferior to the two other algorithms throughout the run for problem h_0. Note that a similar behaviour was observed on the other instances of series H.

### *3.1 Testing Modified MIP-Recombination Operators*

Additional experiments were devoted to a modification of MIP-recombination operator where we also fix all common variables of value 1 and release a

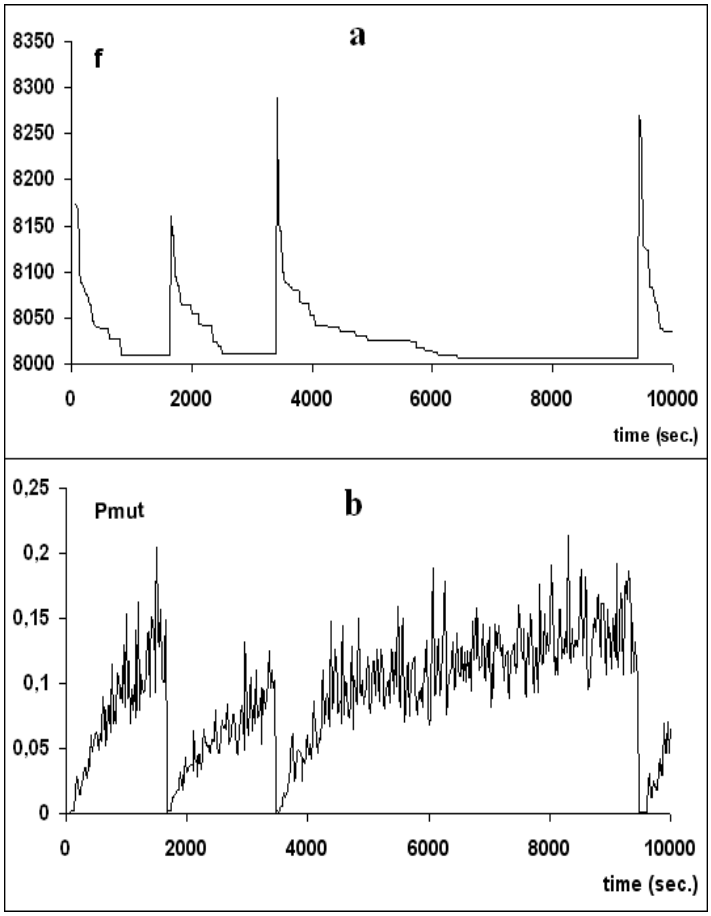


Fig. 1. The best-found solution cost (a), and the mutation probability (b) in each run of GAmipr for problem p_5.

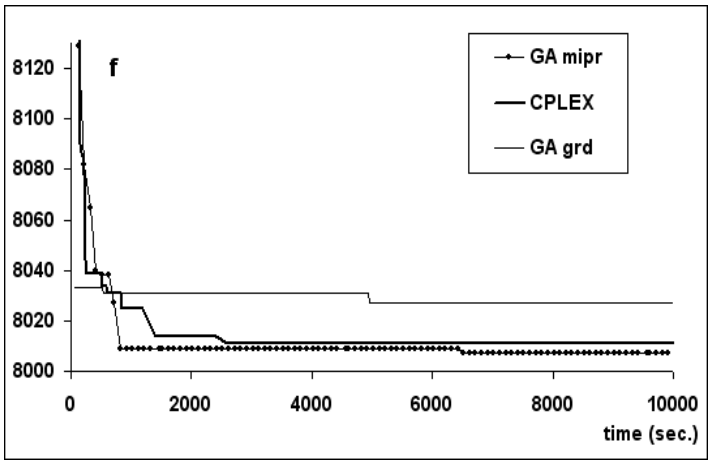


Fig. 2. The best found solution cost on problem p_5 starting from the beginning of the algorithms execution: GAmipr, GAgrd and CPLEX in heuristic mode.

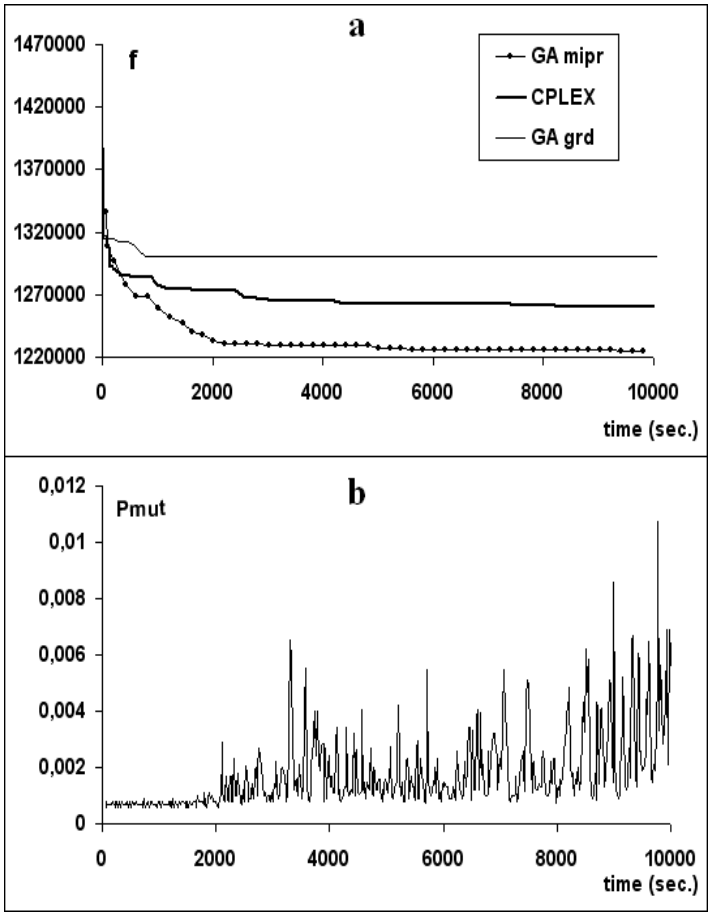


Fig. 3. The best-found solution cost for GAmipr, GAgrd and CPLEX in heuristic mode (a), and the mutation probability (b) in GAmipr on problem h_0.

random subset of the fixed variables, each one being selected with success probability $p_{mut}$. This modification of the operator is symmetric w.r.t. zeros and ones and is closer to the crossover of the standard genetic algorithm (see e.g. a survey of Reeves (1997)), than the MIP-recombination of GAmipr. The experiments showed that this modification reduces the GA efficiency: out of 26 tested instances from different series the best solution found by this modification was better than that of the GAmipr in 4 cases and worse in 12 cases. The average "gap" values increased on series S more than twice and on series H – by a factor of 1.26. It is likely that fixing the variables of common value 1 preserves too many important features of the parent solutions and the population converges too fast.

Another modification was aimed at avoiding generation of duplicate solutions. Here both of the parent solutions $\mathbf{y}^1, \mathbf{y}^2$ and 10 individuals of greatest fitness in the current population $\mathbf{\Pi}^{(t)}$ were cut-off from the set of feasible solutions in the MIP-subproblem, using the Benders cuts. This modification yielded a

statistically significant improvement of solutions cost in the preliminary version of GA with MIP crossover in (Borisovsky *et al.* (2006)). Nevertheless in the GAmipr described in the present paper, the Benders cuts gave no improvement. Probably, the reason is in greater diversity of population in the GAmipr, making the Bendes cuts irrelevant. This is supported by the fact that increasing the tournament size $s$ to 20 and fixing the variables equal to 1, as described above, we observed some positive effect from introduction of the Benders cuts (although the overall performance was still worse than that of GAgrd). Note that the preliminary version of the GA in (Borisovsky *et al.* (2006)) was also tested with $s = 20$ and the variables equal to 1 were partially fixed in the crossover.

We also tested the GAmipr where the CPLEX solver is called with the settings of heuristic mode but this modification led to higher solutions costs on average. So, balance between feasibility and optimality in MIP recombination is more appropriate than feasibility emphasis. Probably, the reason is that our mechanism of adaptive tuning for parameter $p_{mut}$ is aimed at building the recombination subproblems, solvable *to optimality* within the time limit $T_{rec}$.

## 4 Conclusions

The results reported here indicate that integration of a modern branch-and-cut MIP-solver into the recombination procedure can give a valuable operator of a GA. The proposed GA turned out to be superior to a stand-alone branch-and-cut MIP-solver in terms of primary solutions quality and robustness in finding feasible solutions. A simple GA based on a problem-specific Greedy heuristic tends to deliver solutions of lower quality, although it is more robust in finding feasible solutions, compared to the two other algorithms. As noted in Subsection 2.3, the greedy-based GA is also applicable to a more general problem formulation with non-linear cost functions.

The effectiveness of proposed optimal recombination operator is caused by relatively low dimensionality of the optimal recombination subproblems since most of the variables, equal to zero in "parent" solutions, are eliminated from the subproblem. Such behavior can be expected in many other MIP problems, where in the feasible solutions only a small fraction of all variables take non-zero variables. In view of wide applicability of general-purpose MIP solvers, this observation allows us to expect that the MIP-recombination approach may be successfuly extended to a number of other problems. At the same time, in the cases where the optimal recombination may be carried out by provably efficient algorithms (see e.g. Balas and Niehaus (1998); Eremeev (2006)), these eficient algorithms should be preferable to general purpose MIP solvers in recombination.

## 5 Acknowledgements

* Partially supported by INTAS grant 03-51-5501 and Russian Foundation for Basic Research grant 07-01-00410.

The authors would like to thank Lidia Zaozerskaya for providing the generator of random SMP instances of series S and to thank Jens Gottlieb for the information about the best-known solutions to FCTP of series H.

## References

Aggarwal, C.C., J.B. Orlin and R.P. Tai (1997). An optimized crossover for maximum independent set. *Oper. Res.* **45**, 225–234.

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

Barr, R.S., R.S. Glover and D. Klingman (1981). A new optimization method for large scale fixed charge transportation problems. *Oper. Res.* **29**(3), 448–463.

Borisovsky, P., A. Dolgui and A. Eremeev (2006). Genetic algorithms for supply management problem with lower-bounded demands. In: *Proc. of 12th IFAC Symposium "Information Control Problems in Manufacturing 2006" (INCOM'2006)*. Vol. 3. pp. 521–526. Elsevier Science.

Borisovsky, P.A. (2004). Genetic algorithms for a supply management problem. In: *Proc. Intern. Conf. "Systems, Mechanisms and Machines Dynamics"*. pp. 255–258. Omsk. (In Russian).

Chauhan, S.S. and J.-M. Proth (2003). The concave cost supply problem. *Europ. Journ. of Oper. Res.* **148**(2), 374–383.

Chauhan, S.S., A.V. Eremeev, A.A. Kolokolov and V.V. Servakh (2005*a*). Concave cost supply management problem for single manufacturing unit. In: *Supply Chain Optimisation, Product/Process Design, Facility Location and Flow Control* (A. Dolgui, J. Soldek and O. Zaikin, Eds.). pp. 167–174. Springer Verlag.

Chauhan, S.S., A.V. Eremeev, A.A. Romanova and V.V. Servakh (2004). Approximation of linear cost supply management problem with lower-bounded demands. In: *Proc. of Discrete Optimization Methods in Production and Logistics (DOM'2004)*. pp. 16–21. Omsk.

Chauhan, S.S., A.V. Eremeev, A.A. Romanova, V.V. Servakh and G.J. Woeginger (2005*b*). Approximation of the supply scheduling problem. *Oper. Res. Lett.* **33**(3), 249–254.

Cranic, T.G. and M. Gendreau (2002). Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics* **8**, 601–627.

Danna, E., E. Rothberd and C. Le Pape (2005). Exploring relaxation induced neighborhoods to improve MIP solutions. *Math. Program. Ser. A* **102**, 71–90.

Dréo, J., A. Petrowski, P. Siarry and E. Taillard (2006). *Metaheuristics for Hard Optimization: Methods and Case Studies*. Springer.

Eckert, C. and J. Gottlieb (2002). Direct representation and variation operators for the fixed charge transportation problem. In: *Proc. of PPSN 2002*. pp. 77–87.

Eremeev, A.V. (2006). On complexity of optimized crossover for binary representations. In: *Theory of Evolutionary Algorithms. Proc. of Dagstuhl Seminar 06061*. Schloss Dagstuhl, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI).

Eremeev, A.V., A.A. Romanova, V.V. Servakh and S.S. Chauhan (2006). Approximate solution of supply management problem. *Diskretnyi Analiz i Issledovanie Operatcii* **13**(1), 27–39. Series 2 (in Russian).

Gaspero, L. Di, J. Gärtner, G. Kortsarz, N. Musliu, A. Schaerf and W. Slany (2006). The minimum shift design problem: Theory and practice. *Annals of Operations Research*. In press.

Glover, F. and H.D. Sherali (2005). Some classes of valid inequalities and convex hull characterizations for dynamic fixed-charge problems under nested constraints. *Annals of Operations Research* **140**(1), 215–233.

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

Gottlieb, J. and L. Paulmann (1998). Genetic algorithms for the fixed charge transportation problem. In: *Proc. of the 5th IEEE International Conference on Evolutionary Computation*. pp. 330–335.

Höhn, C. and C.R. Reeves (1996). Graph partitioning using genetic algorithms. In: *Proc. of the 2nd International Conference on Massively Parallel Computing Systems*. pp. 31–38. IEEE Computer Society Press.

Johnson, D.S. and Trick, M.A., Eds. (1996). *Cliques, Coloring, and Satisfiability*. Vol. 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. AMS.

Kolokolov, A.A. (1996). Regular partitions and cuts in integer programming. In: *Discrete Analysis and Operations Research*. pp. 59–79. Kluwer Acad. Pbs.

Ortega, F. and L.A. Wolsey (2003). A branch-and-cut algorithm for the single commodity uncapacitated fixed-charge network flow problem. *Networks* **41**(3), 143–158.

Reeves, C.R. (1997). Genetic algorithms for the operation researcher. *INFORMS Journ. on Comput.* **9**(3), 231–250.

Sheng, S., Z. Dechen and X. Xiaofei (2006). Genetic algorithm for the transportation problem with discontinuous piecewise linear cost function. *International Journal of Computer Science and Network Security* **6**(7A), 181–190.

Steinberg, D.I. (1977). *Designing a heuristic for the fixed-charge transportation*

*problem*. Reprints in Mathematics and the Mathematical Sciences. Southern Illinois University at Edwardsville. Edwardsville, IL.

Sun, M., J.E. Aronson, P.G. McKeown and D. Drinka (1998). A tabu search heuristic procedure for the fixed charge transportation problem. *Europ. Journ. of Oper. Res.* **106**, 441–456.

Yagiura, M. and T. Ibaraki (1996). The use of dynamic programming in genetic algorithms for permutation problems. *Europ. Journ. of Oper. Res.* **92**, 387–401.