USE OF GENETIC ALGORITHMS FOR SOLUTION OF THE RECTANGLE PACKING PROBLEM

A. A. Lipnitskii

UDC 519.68

This paper deals with the problem of packing of rectangles in a given rectangle. Genetic algorithms are proposed for solving the problem. Some details of realization of the approach proposed are given and a high efficiency of the method of genetic algorithms is illustrated by two examples.

Keywords: *packing problem, genetic algorithms, packing of rectangles in a given rectangle.*

STATEMENT OF THE PROBLEM

This paper deals with following problem [1]. There is a rectangle P_0 of length l_0 and width w_0. Let a collection of rectangles P_i be also given whose linear sizes are l_i and w_i, where $i \in \overline{1, N}$. We will call the position of a rectangle P_i in P_0 the triple $< x_i, y_i, o_i >$, where x_i and y_i are the coordinates (local in P_0) of the left lower node of P_i and o_i is the orientation of P_i in P_0. We will say that the orientation is direct if the width of the rectangle P_i is oriented along the x-axis and inverse if the length of the rectangle is oriented along the x-axis. In what follows, we will consider that all the positions are inside of the rectangle P_0, i.e., we have

$$\begin{cases} 0 \le x_i, \quad x_i + w_i \le w_0, \\ 0 \le y_i, \quad y_i + l_i \le l_0, \end{cases}$$

for the direct orientation and

$$\begin{cases} 0 \le x_i, \quad x_i + l_i \le w_0, \\ 0 \le y_i, \quad y_i + w_i \le l_0, \end{cases}$$

for the inverse orientation. For two positions, the intersection operation $< x_i, y_i, o_i > \cap < x_j, y_j, o_j >$ is introduced as the intersection of interiors of sets in $\mathbf{R}^2$. We call an arrangement in P_0 a set $R = \{< x_i, y_i, o_i > | i \in I \subset \overline{1, N}\}$ such that, for any two positions $< x_i, y_i, o_i >, < x_j, y_j, o_j > \in R$, we have $< x_i, y_i, o_i > \cap < x_j, y_j, o_j > = \varnothing$ when $i \neq j$.

On the set of arrangements $\mathfrak{R}$, a real-valued function $f: \mathfrak{R} \to \mathbf{R}$ is given. It is required to find an arrangement $R_0 \in \mathfrak{R}$ such that

$$f(R_0) = \max_{R \in \mathfrak{R}} f(R). \tag{1}$$

Example 1. In the capacity of f, we take the cardinality of the set of arrangements $f(R) = |R|$. In this case, the problem is reduced to the determination of the arrangement in which the number of rectangles is maximum.

Example 2. Let $f = \sum\limits_{< x_i, y_i, o_i > \in R} x_i\, y_i$.

This reduced to the determination of the arrangement with the minimal blank area.

If the solution is searched for not on the set of arrangements but on its subset $\mathfrak{R}_0 \subset \mathfrak{R}$, then we have conditional optimization. For example, the set $\mathfrak{R}_0$ can consist of only the arrangements in which the orientation of each position is direct.

In a more general case, some collection of attributes a_{ij} can be assigned to each rectangle P_i. In this case, objective function (1) and the definition of the set $\mathfrak{R}_0$ can depend on a_{ij}.

Institute of Mathematics, National Academy of Sciences of Belarus, Minsk, Belarus, *ert@lipnitsky.org*. Translated from Kibernetika i Sistemnyi Analiz, No. 6, pp. 180-184, November-December 2002. Original article submitted January 5, 2001.

Example 3. Let a price $a_{i1} = c_i$ and a mass $a_{i2} = m_i$ be assigned to each rectangle P_i. It is required to find the most expensive arrangement with some restriction on its mass. In this case, we have

$$f(R) = \sum_{i \in I_R} c_i, \ \mathfrak{R}_0 = \{R \in \mathfrak{R} \mid \sum_{i \in I_R} m_i < C\}.$$

GENETIC ALGORITHMS

The idea of using biological principles of natural selection in artificial systems was proposed some decades ago [2–4]. This method was called evolutionary algorithms or evolutionary computations. At the present time, it is successfully used in various fields such as optimization, automatic programming, operations research, ecology, investigation of social systems, etc. Among the types of evolutionary algorithms are genetic algorithms. Their essence is considered below.

A genetic algorithm is an iterative process over a population that consists of individuals and is of constant size. Each individual is represented by a symbol string, which is called a genome. As a rule, the symbols 0 and 1 are used (binary coding). A given string codes information on a possible solution of a formulated problem from a given search space. Genetic algorithms are used in the cases where the search space that contains all possible solutions is too large for an exhaustive search. The scenario of a standard genetic algorithm is as follows. The initial population of individuals is generated randomly or heuristically. At each evolutionary step, the individuals from the current population (generation) are estimated by a criterion specified in advance (a utility function). The next generation is formed with the help of selection of individuals of the current generation according to their utility. The choice of a given number of individuals with probability proportional to the value of the utility function is among the simplest variants of selection. Thus, the "better" is an individual, the better is his chance to be involved in the next generation. On the other hand, the individuals with a small value of the utility function will not pass, most likely, to the next step of evolution.

Selection in itself is insufficient for generation of new individuals in a population, i.e., for determination of a new point in the search space. There exist two methods of solution of this problem, namely, crossover and mutation. Two individuals (parents) from the current population are crossed with the help of exchange of segments of their genomes. Two new individuals called descendants come into existence in this case. In the simplest form of crossover, substrings of their genome are exchanged at a randomly chosen break point. The choice of candidates for crossover is, as a rule, also based on the values of the utility function, i.e., the "best" individuals have more chances of engendering descendants than the "worst" ones. The individuals of the generation being formed mutate by changing a symbol in their genome strings with a sufficiently small probability.

The selection operator makes it possible, in passing from generations to generation, to preserve individuals with sufficiently high values of the utility function in a population. At the same time, the crossover operator provides the motion of a population to the most promising domain of the search space. The mutation operator makes it possible to avoid the cycling of an algorithm near a local optimum. It should be noted that a genetic algorithm is a stochastic iterative process, which, in general, does not guarantee the convergence to the best solution. Therefore, a termination criterion must be chosen on the basis of a concrete statement of the problem being considered. The most used criteria of termination of evolution are the obtainment of some value (specified in advance) of the utility function by individuals or the alternation of a definite number of generations.

REALIZATION

Let us consider the use genetic algorithms for solving the problem of optimization of arrangements that is formulated above (see also [5]).

Coding of Information on an Arrangement in a Genome. Denote by O the set of all orientations, i.e., we have $O =$ {"direct"; "inverse"}. Let $\mathfrak{U} \subset \mathfrak{R} \times I_{[1, N]} \times O$, and we have $(R, i, o) \in \mathfrak{U}$ if R is an arrangement that does not contain the rectangle with a number i. Then a mapping $A: \mathfrak{U} \to \mathfrak{R}$ is called a unit arrangement algorithm if $A(R, i, o) = R$ or $A(R, i, o) = R \cup \{< x_i, \ y_i, o >\}$. Thus, a unit arrangement algorithm calculates the coordinates of a rectangle and arranges it inside P_0 and if the rectangle cannot be arranged, then it preserves the current arrangement. Denote by $\mathfrak{A}$ the set of all the used unit arrangement algorithms. An example of a unit arrangement algorithm can be the determination of the minimum value of the ordinate of the rectangle being arranged and also, in the event of an ambiguity, the minimum value of its abscissa such that the rectangle is completely arranged in P_0 and does not intersect with rectangles arranged in P_0.

The entire genome is divided into N equal parts (genes) $G = < G_1, \dots, G_n >$ according to the number of the rectangles being arranged. Each gene G_i is a triple $G_i = < a_i, o_i, n_i >$, where a_i is a unit arrangement algorithm, o_i is the value of orientation, and n_i is the number of a rectangle and, at the same time, $n_i \neq n_j$ when $i \neq j$. Let G be a genome; then an arrangement $R = R_N$ can be obtained from it by the recurrent rule

$$\begin{cases} R_0 = \varnothing, \\ R_i = a_i(R_{i-1}, n_i, o_i), \ i = \overline{1, N}. \end{cases}$$

Initial Generation of a Genome. A genome G is generated at random. In other words, for each $i \in \overline{1, N}$, a_i from the set of all the unit arrangement algorithms, o_i from the set of orientations, and n_i from the set $I_{[1, N]}$ are randomly chosen and, at the same time, all the n_i are different.

Crossover Operation over Two Genomes. Let there be two genomes G' and G''; the crossover operator makes it possible to obtain a new genome G by the following rule: $G = < G'_1, \dots, G'_m, G''_{m+1}, \dots, G''_N >$. To choose the break point that is specified by a number m, we will use the following rule. Let us consider a sequence of natural numbers $1 \leq m_1 < m_2 < \dots < m_l \leq N$ such that the initial parts of the genomes $< G'_1, \dots, G'_{m_i} >$ and $< G''_1, \dots, G''_{m_i} >$ have the same number of rectangles of each type for any $i \in \overline{1, l}$. We will say that two rectangles belong to one type if their lengths, widths, and also collections of attributes coincide. Then m is randomly chosen from $m_1, m_2, \dots, m_l$. Thus, G actually is the genome for some arrangement.

Operation of Mutation of a Genome. A gene G_i is randomly chosen from the genome G. In this gene, we randomly change the current unit arrangement algorithm a_i or orientation o_i.

Let the entire collection of rectangles P_i be subdivided into M types, and let each type consist of N_i elements, $\sum_{i=1}^{N} N_i = N$. The cardinality of the search space is specified by the formula

$$2^N |\mathfrak{A}|^N \frac{N!}{N_1! N_2! \dots N_M!}. \tag{2}$$

EXAMPLES

Let us illustrate the functioning of the above algorithm by two concrete examples. In Tables 1 and 2, the descriptions of geometric parameters of the rectangles being arranged are given. The size of the initial rectangle equaled 210×150. In the capacity of the utility function, the number of the rectangles arranged was used. The termination criterion was as follows. If the optimum was reached at some moment, i.e., all the rectangles were arranged, then the program came to an end. If the optimum was not reached during some minimal number $C_{\min}$ of cycles, then the program searched for it during at least C_{win} cycles. If a solution that was better than all the preceding ones was found during this interval of time, then the program additionally searched for the optimum during C_{win} cycles, etc., but no longer than the maximum number of cycles $C_{\max}$ specified in advance. Otherwise, if the solution was not improved during the last C_{win} cycles, the program came to an end with the best solution found. In all cases, the population size equaled one thousand individuals and each next generation was formed on the basis of 20% of selection, 70% of coupling, and 10% of random strategies. During selection and coupling, individuals were randomly chosen proportionally to values of their utility functions. The probability of a mutation for each individual on each cycle equaled 1/1000. The set of unit arrangement algorithms consisted of one element, namely, each rectangle was arranged so that the ordinate of its left lower angle was minimal among all possible positions and, in the event of an ambiguity, the minimality of the abscissa value was required. In this case, the lower side of the corresponding rectangle was in complete contact with the boundary of the arrangement domain or with the sides of the rectangles arranged earlier.

For each example, ten computational experiments were carried out. In Table 3, the serial numbers of the generations are shown for which the optimum was achieved.

As it is obvious from the table, the average number of points from the search space that were considered by the algorithm did not exceed 667 000 in the first case and 5 697 000 in the second case. The cardinalities of search spaces computed by formula (2) are approximately equal to $6.4 \cdot 10^{62}$ and $3.3 \cdot 10^{35}$, respectively. This illustrates a high efficiency of the method proposed.

TABLE 1

Length	Width	Number
10	10	30
20	20	7
20	10	3
30	10	4
30	30	3
30	20	7
40	40	8
40	30	2
60	30	1
Total		65

TABLE 2

Length	Width	Number
10	10	15
20	20	7
20	10	1
30	10	4
30	30	3
30	20	1
40	40	3
40	30	1
150	60	1
150	50	1
Total		37

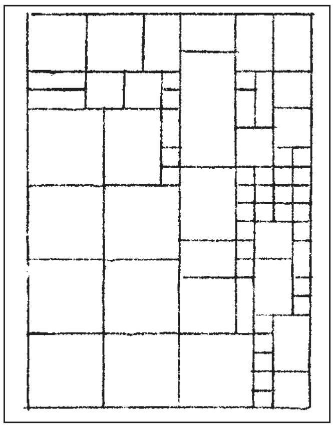
Fig. 1

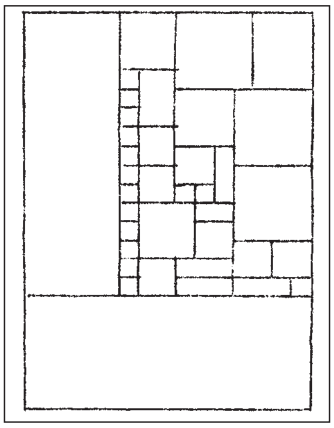
Fig. 2

TABLE 3

Example	Number of Generation										Average
	1	2	3	4	5	6	7	8	9	10	
1	365	1214	3	779	4	280	116	89	3252	568	667
2	2023	6763	8279	7008	2257	7452	6460	6478	8457	1793	5697

In Figs. 1 and 2, the schemes of arrangements found with the help of the program that realize this genetic algorithm are given.

REFERENCES

1. Yu. G. Stoyan and S. V. Yakovlev, Mathematical Models and Optimization Methods in Geometric Design [in Russian], Naukova Dumka, Kiev (1986).
2. J. Y. Holland, Adaptation in Natural and Artificial Systems, Univ. of Michigan, Michigan (1975).
3. D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Boston (1989).
4. Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin (1996).
5. A. A. Lipnitskii and B. Ye. Neselovskii, “Optimization of arrangements on a plane with the help of genetic algorithms,” in: Proc. 8th Byelorus. Math. Conf., Vol. 3, Minsk (2000).