The Application of GLS Algorithm to 2 Dimension Irregular-Shape Cutting Problem

Luiza Budzyńska and Paweł Kominek*

Institute of Computing Science, Poznan University of Technology,
Piotrowo 3a, 60-965 Poznan
{Luiza.Budzynska,Pawel.Kominek}@cs.put.poznan.pl

Abstract. This paper describes the application of the Genetics Local Search Algorithm (GLS) to the 2 Dimension irregular-shape Cutting Problem. We describe different recombination operators used to generate feasible solutions, as well as an algorithm that can be used to rotate figures in this problem. In our case, each offspring resulting from a recombination is a starting point for local optimization. The resulting solutions are permutations of the figure labels (i.e. sequence-coding). One of the distinctive features of our method is a specific representation of figures, in which a 32 bit binary vector is transformed into an integer value. Figures obtained in this way are then placed on the strip (encoded in the same way) in order to obtain the final solution. The paper includes results of computational experiments demonstrating the efficiency of the proposed approach.

1 Introduction

The Genetic Local Search (GLS) is a metaheuristic algorithm that combines genetic (evolutionary) algorithms with local optimization. Other frequently used names are Memetic Algorithms or Hybrid Genetic Algorithms [11]. It is well known that the way in which the evolutionary algorithms are adapted to a particular problem may have a crucial influence on its performance. The paper considers 2-Dimension Cutting (2-DC) problem [1] of minimizing the stripe length. Because of high computational complexity [7] of the problem the use of evolutionary algorithm with local search (GLS) based on significant features is proposed [4, 9]. Heuristics based on the GLS scheme often prove to be extremely efficient in combinatorial optimization [6, 9].

It is quite difficult to track the single origin of GLS. An important element to adopt GLS to a particular problem is the representation of data that we use to create its solution. Frequently the representations come from intuition [10] but a successful representation should not only be natural, but it should also be efficient from the point of view of local search. Traditional genetic algorithms use binary coding of solutions [8] but for many problems, such coding is not natural. Recently, dataset are often encoded with some specialized data structures [5].

* This work was supported by grant no 3 T11F 004 26 from the State Committee for Scientific Research.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 1241–1248, 2004.

In this paper, we propose a dynamic structure to remember free pixels on the strip and we propose to use an array of integer values to minimize the amount of used memory and improve efficiency of combinatorial optimization. After this introduction the 2-DC problem to which we apply our approach is described in the second section. In the third section, we describe in detail the encoded solution and the figure representations. Next, we describe the adaptation of the dynamic structure to the GLS algorithm to solve the 2-DC problem. Computational experiments are described in the fifth section. In the sixth section, conclusions are presented.

2 Problem Definition

The 2-Dimension Cutting (2-DC) problem [1] considered in this paper consists in minimization of used strip length. The strip is a rectangle with unlimited length and determined width. To find a solution we use a procedure called Bottom-Up-Left-Justified, in which the first figure to be put is placed in left-up corner of the strip. The figures are moved pixel-by-pixel across the strip, first down and then if it not possible to the right, until a free location is found. None of the figures can overlap. The figures are any polygons, with possible holes made of other polygons. Depending on the instance of the problem, the rotations of figures are allowed or not.

3 Coding for the 2-DC Problem

3.1 Solution Coding

In our method, every solution is represented as a sequence of numbers, called a sequence list. These sequences are the identifiers of the figures, which are ordered according to procedure Bottom-Up-Left-Justified described above (Fig. 1 shows an example of 20 figures).

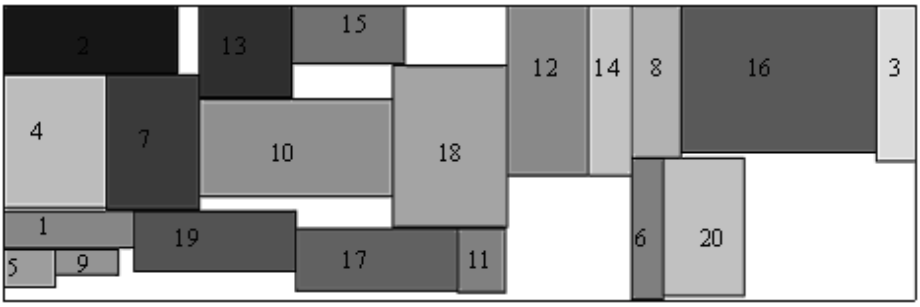

Fig. 1. Example of sequence coding: [2-4-1-5-9-7-9-13-10-15-17-18-11-12-14-8-6-20-16-3]

3.2 Figure Representation

The first implementation of this problem using binary and integer maps was created for the requirements of a vector machine [12]. It seemed that such encoding would not be effective on machines without vector registers. In spite of all we decided to verify this methodology on PC computers. In the considered problem all figures are polygons. Possible holes in the polygons are also represented as polygons. During the conversion of the representation from vectors (vertices) to binary map representation we approximate sides of figures with segments of line. The polygons obtained in such way are greater than or equal to the input figures. This results from the resolution of the binary map. Next we fill the polygons' interiors using the following algorithm:

```
ALGORITHM_1 FILL_THE_FIGURE
BEGIN
   For the considered polygon take all pairs of
   vertices and compute a - the slope of the line of
   its sides, and b - the intercept (y=ax+b);
   From all identified sides remove those, which are
   parallel to the x-axis (a=0);
   Sort all sides preserved in the preceding step
   according to the x-coordinate and then y-coordinate
   of their vertices;
   Take each pair of the preserved sides and
   horizontally fill the area between them;
END.
```

This algorithm is one of many we have tested for filling the interiors of polygons. It proved fast and accurate in that it always correctly filled polygons, even those having holes. We used the binary map representation for the figures, but the computational times achieved were not fully satisfying. Therefore, we converted the binary map into an integer map [4].

4 Implemented Algorithm and Data Structures

To solve the 2-DC problem we used the Genetic Local Search algorithm in the following version:

```
ALGORITHM_2 GENETIC_LOCAL_SEARCH
BEGIN
  P:=0;
  FOR i:=1 TO N DO
         BEGIN
           Construct a new feasible solution x;
           Apply local search to x obtaining x';
           Add x' to P;
         END
  REPEAT
```

```
      Draw at random two solutions x and y from P ;
      Recombine x and y obtaining z;
      Apply local search to z obtaining z';
      if z' is better than the worst solution in P and
      different to all solutions in P then Add z' to P
      and delete the worst solution from P;
      P:=P+1;
    UNTIL stopping criterion is met
  END.
```

where,

```
    P   - population id,
    N   - size of population,
```

4.1 Proposed Recombine Operators

Several operators take advantage of the sequence lists. One of the proposed recombination operators finds pairs of figures adjacent to each other and common to two parent solutions and places them in their offspring, and completes the offspring with randomly selected figures. Another proposed operator finds pairs or triplets of figures (not necessarily adjacent) following the same order in parent solutions and places them in the offspring, completing it with randomly selected figures. The last proposed operator constructs an offspring that has the same features in common with its parents as the two parents have in common with each other.

All operators were tested on different instances [3]. The results indicate that the use of GLS with the last recombination operator indeed improves both the quality of the new local optima and reduces the CPU time needed to reach them.

4.2 Identification of Free Pixels on the Stripe

Initially, the strip represented by array of values contains only empty values. Whenever an insertion of a figure is evaluated, the appropriate bits on the strip are tested. Each time the figure-placing procedure starts from the first row and the first column of the array. To speed up the procedure we proposed to additionally define a vector of columns including indices of **f**irst **f**ree **r**ow in each column (denoted as FFR) or an additional array including **a**ll indexes of **f**ree **r**ow [3] (denoted as AFR). However, this approach did not reduce the computational time of GLS.

4.3 Rotation of Figures

To rotate figures we perform the rotation of coordinates at an a priori given angle φ (fig. 2 shows an example).

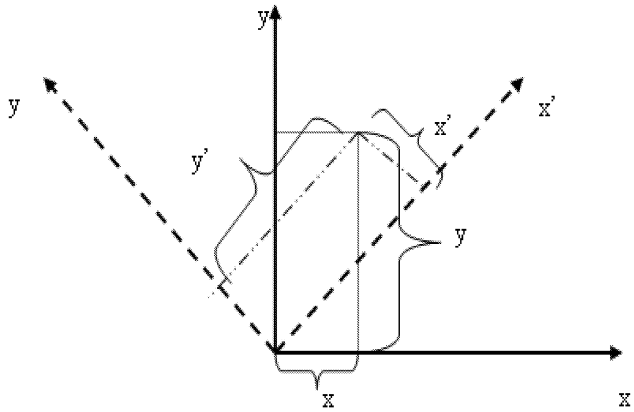

Fig. 2. Rotation of coordinates

To calculate new coordinates of figure vertices x' and y' we use the following formula (1)

$$x = x'\cos(\varphi) - y'\sin(\varphi),\ y = x'\sin(\varphi) + y'\cos(\varphi)$$
$$x' = x\cos(\varphi) + y\sin(\varphi),\ y' = -x\sin(\varphi) + y\cos(\varphi) \tag{1}$$

5 Computational Experiments

At first our implementation used arrays to encode data, then we applied vector containers from STL library. During the first experiment on PC, when array structures were used, computational time was satisfying. This encouraged us to perform further research aiming at finding the best appropriate structures. The proposed vector container resembles a C++ array in that it holds zero or more objects of the same type. The vector container can contain elements of any types. The STL library implements different methods in vector container, e.g.: insert, add, clear, resize, which are used in our implementation. In the experiments, we used seven different instances of the problem. The instances differed in the sets of figures to be put on the strip and the angle of rotation. For the instances denoted as “43a shape” and “10a pros” the rotation of 90° and 180° was allowed for each figure. In [5] Oliveira at all presents the best known layout for “43 shape” and “43a shape”.

The experiments were performed on a PC with Pentium 733 MHz processor. We have noticed that an increase of the running time did not influence the quality of results significantly. Table 1 presents comparison of creation times of one randomly selected solution for binary (BM) and integer map (IM) representation of the figures.

Table 2 contains experimental results obtained with a number of different methods described in section 4.2. In all cases, populations of size 40 were used.

The proposed methods are also compared to a standard GLS algorithm without any additional structures (see Table 3). In addition, we test the best kind of GLS (i.e. the one with FFR structure) on several instances (see Table 4). All the algorithms shared a common code, for the most part.

Table 1. Creation time of solution

Name of instance	Fitness function [pixel]	Time [s]
10 pros (BM)	720	2,6
10 pros (IM)	720	1,6
23 pros (BM)	650	271,9
23 pros (IM)	650	4,8
45 pros (BM)	1490	4,9
45 pros (IM)	1490	0,6

Table 2. Average of 100 solutions for GLS algorithm with FFR and AFR structure for different instances

Name of instance	AFR structure				FFR structure			
	Fitness function [pixel]		Time [s]		Fitness function [pixel]		Time [s]	
10 pros (BM)	809	±97,7	5,5	±4,0	804,4	±73,3	2,3	±0,87
23 pros (BM)	713,6	±76,4	200,2	±36,4	672	±38,8	165,4	±59,7
45 pros (BM)	1582,4	±48,6	2,2	±1,4	1596,9	±49,6	2,2	±0,9

Table 3. Average of 100 solutions for GLS algorithm without any additional structure for different instances

Name of instance	Fitness function [pixel]		Time [s]	
10 pros (BM)	792,4	±60,1	3,7	±3,7
23 pros (BM)	671,6	±79,7	201,3	±46,5
45 pros (BM)	1572,4	±44,9	2	±0,8

Table 4. Average of 100 solutions for GLS algorithm without any additional structure for different instances

Name of instance	Fitness function [pixel]		Time [s]	
43 shape	67,6	± 1,1	70,6	±24,6
43a shape	63,6	±0,5	165,7	±12,8
10 pros	682	±17,9	33,5	±10,1
10a pros	626	±15,8	50,9	±5,9
23 pros	571,7	±10,7	176,44	±11,3
45 pros	1473,57	±19,8	31,3	±1,3
20 polygon	1001,4	±8,3	123,7	±5,4

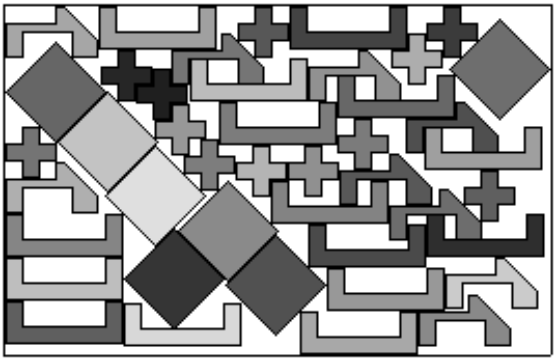

Fig. 3. The best known layout for instance “43 shape” obtained with GLS

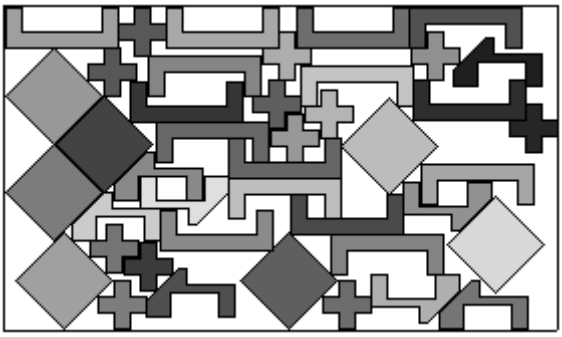

Fig. 4. The best known layout for instance “43a shape” obtained with GLS

Figures 3 and 4 present the best layout for “43 shape” and “43a shape” obtained by GLS algorithm solving 2-DC problem where figures are transformed to the integers map and the first bits on the strip are known.

6 Conclusion

In this research we have tested two figure representations for the 2-DC problem using GLS algorithm. The first of them is a binary map representation of the figure denoted as (BM) and the second is an integer map denoted as (IM).

The results of the experiments prove that the GLS based on integer map representation outperforms the GLS based on binary map representation.

Another interesting issue to point out is the use of a dynamic structure to remember free pixels on the strip. The results indicate that the GLS algorithm using additional dynamic structure does not give better results than GLS without such structure.

Software for 2-DC problem is allowed under location:

http://www-idss.cs.put.poznan.pl/~pkominek

References

1. J. Blazewicz, P. Hawryluk, R. Walkowiak, Using a tabu search approach for solving the two-dimensional irregular cutting problem, Annals of Operations Research, Vol 41, pp 313-327, 1993.
2. P. C. Gilmore, R. E. Gomory, Multistage Cutting Problems of two and more dimensions, Operations Research 13:94--119, 1965.
3. L. Dolata, P. Kominek: An evolutionary algorithm using expert's knowledge for 2-dimension irregular-shape cutting problem, Proceedings of the AI-METH 2002 - Symposium on Methods of Artificial Intelligence, Gliwice 13-15.11.2002, pp. 147-150.
4. L. Budzyńska, P. Kominek, Influence of given representation on Performance of an evolutionary algorithm, Proceedings of the AI-METH 2003 - Symposium on Methods of Artificial Intelligence, Gliwice 5-7.11.2003, pp. 135-138.
5. J.F. Oliveira, A.M. Gomes, J.S. Ferreira, TOPOS - a new constructive algorithm for nesting problems, OR Spektrum 22, 263–284, 2000.
6. B. Freisleben, P. Merz, A genetic local search algorithm for travelling salesman problem., In H.-M. Voigt, W. Ebeling, I. Rechenberg, H.-P. Schwefel (eds.), Proceedings of the 4th Conference on Parallel Problem Solving fram Nature- PPSN IV, pp. 890-900, 1996.
7. M. Garey, D. Johnson , Computers and intractability: A guide to the theory of NP-completeness, Freeman, San Francisco, Calif, 1979.
8. D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, 1988.
9. A. Jaszkiewicz, P. Kominek, Genetic local search with distance preserving recombination operator for a vehicle routing problem. European Journal of Operational Research, 151/2, 352-364, 2003.
10. Z. Michalewicz, Genetic algorithms + Data structures = Evolution programs, Springer Verlag, Berlin Heidelberg, 1992.
11. N. J Radcliffe., P. D. Surry, Formal memetic algorithms, in: T. Fogarty (Ed.), Evolutionary Computing: AISB Workshop, Springer-Verlag), 1994.
12. L. Dolata, Przetwarzanie wektorowe dla problemu rozkroju, Praca magisterska, Politechnika Poznańska, Poznań, 2000.