A Comparison of Different Recombination Operators for the 2-Dimensional Strip Packing Problem

Carolina Salto
Facultad de Ingeniera, Universidad Nacional de La Pampa
Calle 110 esq. 9, General Pico, La Pampa (6360), Argentina
saltoc@ing.unlpam.edu.ar

and

Enrique Alba - Juan M. Molina
E.T.S.I. Informática, Universidad de Málaga
29071 Málaga, España
{eat,jmmb}@lcc.uma.es

Abstract

In this paper, the three-stage two-dimensional rectangular strip packing problem is tackled using genetic algorithms. A new problem dependent recombination operator, called best inherited levels recombination (BIL), is introduced. A comparison of its performance is carried out with respect to four classical recombination operators. A complete study of the influence of the recombination operators on the genetic search, including the trade-off between exploration and exploitation in the search process, is presented. The results show that the use of our specialized BIL recombination outperforms the others more generic on all problem instances for all the metrics tested.

Keywords: Strip Packing, Recombination, Genetic Algorithms.

1 INTRODUCTION

The two-dimensional strip packing problem (2SPP) is present in many real-world applications such as in the paper or textil industries, and each of them can impose different constraints and objectives. Typically, the 2SPP consists of a set of M rectangular pieces, each defined by a *width* w_i and a *height* h_i, with $i = 1...M$, wich have to be packed in a larger rectangle with a fixed width W plus an unlimited length, designated as the *strip*. The search is for a layout of all the pieces in the strip that minimizes the required strip length and, where necessary, takes additional constraints into account. This problem is similar to the one of cutting the pieces out of the strip minimizing the consumed strip.

In the layout, the pieces have to be packed with their vertical edges parallel to the vertical edge of the strip (corresponding to orthogonal cuts). Additionally, another constraint is included in the problem: guillotine cuts, i.e. the pieces have to be cut going from one border straight to the opposite side. The packing pattern is created by recursively bi-partitioning the strip with straight lined: guillotineable cuts. Each of these cuts splits a rectangular area within the strip into two rectangular pieces. Additional to this constraint n-staged guillotine cuts can be imposed in some cases. Here there are a limitation in the number n of stages in which the cuts are made. Each stage is only able to perform either horizontal or vertical cuts but not both, and pieces having passed a stage may not be put back to a previous stage. These conditions limit the nesting of horizontal and vertical cuts. The corresponding packing is built as a series of *levels*, each piece being placed so that its bottom rests on one of these levels. The first level is simply the bottom of the strip. Each subsequent level is defined by a horizontal line drawn through the top of the tallest piece on the previous level. Particularly in this work, we focus on three-stage cuts. In the first stage, horizontal cuts (parallel to horizontal edge of the strip) are performed to the strip producing an arbitrary number of *levels (stripes)*. In the second stage, those levels are processed by vertical cuts generating an arbitrary number of so-called *stacks*. The third stage produces the final elements (and waste) from the stacks performing only horizontal cuts.

The 2SPP is NP-hard (see the work of Hopper and Turton [11]). A few exact approaches for this problem are known. Martello et al. [17] developed a branch-and-bound method while Fekete and Schepers [6] proposed a general framework for the exact solution of more-dimensional packing problems.

Regarding surveys of meta-heuristics in the literature, Hopper and Turton [9, 11] review the existing approaches to solve 2D packing problems. The focus is hereby on the analysis of the methods involving genetic algorithms, simulated annealing (SA), tabu search (TS), and artificial neural networks are also used although. The authors give a structured overview regarding regular and irregular packing, guillotine and non-guillotine packing. They conclude that evolutionary algorithms are the most widely investigated meta-heuristics in the area of cutting and packing. Also, due to the lack of benchmarking, it is difficult to decide which method is better suited to approach packing problems. Lodi et. al [16] in their work consider the two dimensional strip packing problem, discussing mathematical models. Moreover the authors survey lower bounds, classical approximation algorithms, recent heuristic and meta-heuristic methods and exact enumerative approaches. The relevant special cases where the items have to be packed into rows forming levels are also discussed in detail.

Studies found in the literature applying meta-heuristics to deal with the SPP are described in Table 1. The emphasis is placed in the constraints optionally included in the problem such as: guillotine constraint (guillotine (G) or non-guillotine (NG) cuts), the orientation constraint

Table 1: Overview of papers dealing with 2D strip packing problem.

Papers	Problem Characteristics			Meta-heuristic	Observations
	cuts	orientation	placement routine		
Bortfeldt [4]	G-NG	O-R	level packing algorithms	GA	No encoding of solutions: fully defined layouts are manipulated by means of specific operators. Heuristic for post-optimizing of layouts (move pieces in a layer into an adjacent layer and to use previously free space). Includes a omparison with many works.
M-Valenzuela et al. [19]	G	R	level packing algorithms and the Split algorithm	GA	Normalised postfix representation. Standard genetic operators. Data sets of various sizes with a variety of characteristics.
Hopper and Turton [10]	NG	R	botton left algorithm	GA, SA and naïve evolution	Packing pattern represented by a permutation. Traditional BL algorithm and an improved BL capable of filling existing gaps.
Liu and Teng [15]	NG	R	improved BL-algorithm	GA	Packing pattern represented by a permutation.
Jakobs [13]	NG	R	botton left algorithm	GA	Permutation representation. The algorithm is improved by combination with deterministic methods.
Kröger [14]	G	R		GA (sequential and parallel verions)	Slicing tree. structure. Hill climbing as special operator.
Hwang et al. [12]	G	R		3 GAs	Polish representation plus special operators; the bin width is incorporated as a penalty function. Ordered list of items and different packing algorithm: a level oriented first-fit algorithm and a level oriented best-fit algorithm.

(fix orientation (O) or the pieces can be rotated (R)). Also the kind of meta-heuristic used, the placement routine to decode a solution to a complete layout (if necessary) and some additional observations (such as the representation adopted to encode a packing pattern) are detailed.

From all the previous approaches we will use in this article evolutionary algorithms [2, 18], in particular genetic algorithms (GAs), as the general driving force to locate the region in which a solution of minimum length is located. GAs deal with a population of tentative solutions, each of them encodes a problem solution on which genetic operators (such recombination of partial solutions) are applied in an iterative manner to progressively compute new solutions of higher quality.

In this paper a hybrid approach is used to solve the two-dimensional strip packing: a GA is combined with a heuristic placement routine. The GA is used to determine the sequence in which the pieces are to be cut. Then a second algorithm is necessary which determines the layout of the pieces onto the object (placement heuristic), respecting guillotine cuts and 3-stage level packing. Moreover, the algorithm uses a special built-in recombination incorporating problem specific knowledge such as information of the pieces layout. The main goal of this paper is to find an effective operator to solve larger problems than the ones found in the literature at present, and to quantify the effects of including these operations into the algorithms. Furthermore, we perform an empirical study where we compare the performance of the new recombination operator with four other classical recombination operators.

The organization of this paper is as follows. The components of the evolutionary approach are described in Section 2. In Section 3 we review the studied recombination operators and present a detail description of the new recombination operator. In section 4, we explain the parameter settings of the algorithms used in the experimentation. Section 5 reports on the algorithm performances, and finally, in Section 6, we give some conclusions and analyze future search directions.

2 GENETIC ALGORITHM FOR THE 2SPP

In this section we present a GA for solving the two-dimensional strip packing problem used in this work. In Algorithm 1 we can see the structure of a GA in which we will now explain the steps for solving our cutting tasks. The algorithm creates an initial population $P(0)$ of μ solutions to the cutting problem in a random way, and then evaluates these solutions. The evaluation uses a placement (ad hoc or heuristic) algorithm to arrange the pieces into sheets to construct a feasible cutting pattern. After that, the population goes into a cycle where it undertakes evolution, which means the application of recombination and mutation operators, to create λ offspring. Finally, each iteration ends by selecting μ individuals to build up the new population from the set of $\mu + \lambda$ existing ones. In this study, the stopping criterion for the cycle is to reach a maximum number of evaluations ($max\_evaluations$). The best solution is identified as the best individual ever found which minimizes the number of needed sheets. Details of implementation are explained in following subsections.

Algorithm 1 Genetic algorithm

```
GA
t = 0; {current generation}
initialize(P(t));
evaluate(P(t));
while (not max_evaluations) do
   P'(t) = evolve(P(t)); {recombination and mutation}
   evaluate (P'(t));
   P(t+1) = select new population from P'(t) ∪ P(t);
   t = t + 1;
end while
```

2.1 Representation

In order to develop a GA for this problem the first step is to develop an adequate encoding of the cutting patterns that will represent a solution to the problem. We encode a cutting pattern in a chromosome as a sequence of pieces that defines the input for a layout algorithm, and pieces are represented by natural numbers. Therefore, a chromosome will be a permutation $\pi = (\pi_1, \pi_2, ..., \pi_M)$ of M natural numbers. For example, let us consider the following problem with eight pieces ($M = 8$), and a strip of width W and an unlimited length. A possible cutting pattern is $\pi = (2\ 4\ 1\ 5\ 6\ 3\ 8\ 7)$, where 2 stands for piece 2, 4 for piece 4, 1 for piece 1, and so on. The chromosome gives an order for considering the pieces, which is used afterwards by a layout algorithm. This algorithm will arrange the pieces to be cut on the strip to build a cutting pattern (see Figure 1). The GA will devise the best possible permutation so that the layout algorithm finds an optimal cutting pattern, which minimizes the required length of the strip.

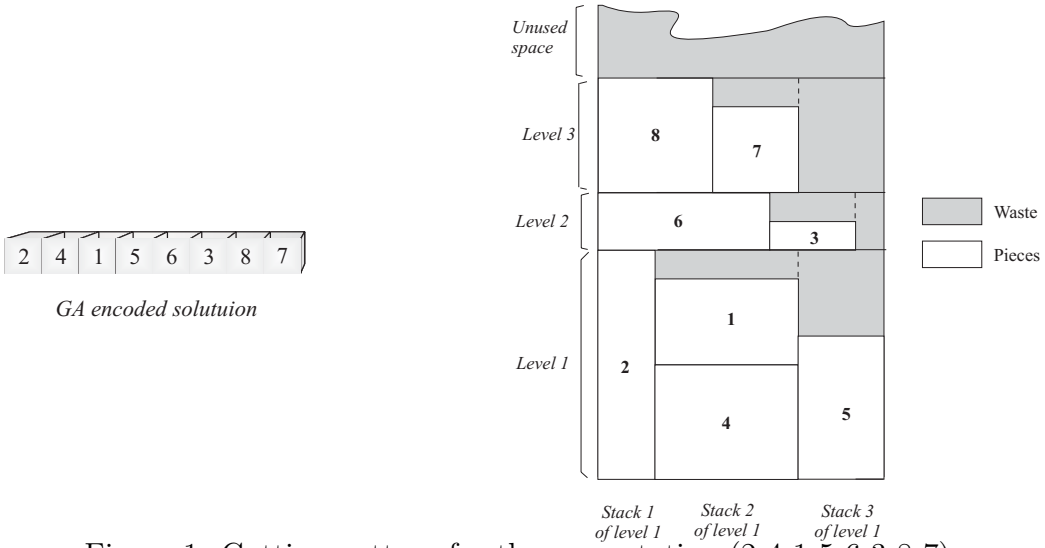

Figure 1: Cutting pattern for the permutation (2 4 1 5 6 3 8 7).

Algorithm 2 MNFH assignment process.

MNFH(π: vector, M:integer, W:integer)
initialize counters:
  $i = 1$; {piece to be placed}
initialize *strip*:
  $strip.length = 0$;
while ($i \leq$M) **do**
  initialize_level(π_i, l);
  $strip.length = strip.length + l.length$;
  while ($i \leq$ M **and** feasible π_i in l) **do**
    initialize_stack(π_i, s, l);
    while ($i \leq$M **and** feasible π_i in s) **do**
      push_stack(π_i, s, l);
      $s.length = s.length + length(\pi_i)$;
      $l.waste = l.waste$ - area(π_i);
      $i = i + 1$;
    end while
  end while
end while
set l to be the last level;
return $strip.lenght, l$;

initialize_level(π_i:piece, l: level)
$l.length = length(\pi_i)$;
$l.remWidth = W$;
$l.waste = strip.area$;

initialize_stack(π_i:piece s: stack, l:level)
$s.width = width(\pi_i)$;
$s.length = 0$;
$l.remWidth = l.remWidth - width(\pi_i)$;

push_stack(π_i :piece, s: stack, l: level)
if $l.length < s.length + length(\pi_i)$ **then**
  $waste = l.waste + (s.length + length(\pi_i) - l.length) \times W$;
  if ($waste-$area(π_i) $\leq l.waste$) **then**
    $l.waste = waste$;
    $l.length = s.length + length(\pi_i)$;
    $strip.length = strip.length + (s.length + length(\pi_i) - l.length)$;
  end if
end if

2.2 Heuristic for Placing Pieces

In order to generate a solution to the three-stage two-dimensional strip packing problem, a modified next-fit decreasing height heuristic (NFDH) is used here (in the following referred as *MNFH*) like the one used in [21, 22]. This heuristic gets an ordered list of all the pieces as its input. Its pseudocode is given in Algorithm 2. The cutting pattern is constructed level by level in a greedy way, i.e., once a new level is started, previous levels are never reconsidered. The same policy is applied to stacks.

The algorithm starts with the first piece π_1 of the given vector $\pi = (\pi_1, \pi_2, ..., \pi_M)$, where M is the number of pieces. Variable *strip* represents the strip, variable l stands for the current level, and variable s stands for the current stack. The strip is initialized as well as its first level with length $l.length = lenght(\pi_1)$, and the first stack inside that level with width $s.width = width(\pi_1)$. Once a level is initialized, the next piece, π_i, of π starts the second stack of the current level if the following conditions are met: (a) $length(\pi_i)$ does not exceed the length of the level, $l.length$, (b) $width(\pi_i)$ does not exceed the current remaining width of the level, $l.remWidth$. If the following pieces have width equal to $width(\pi_i)$ they are stacked one under

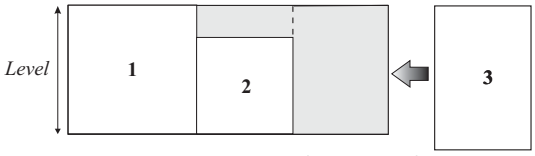

Figure 2: Level example.

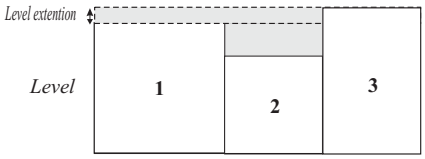

Figure 3: Level extension.

the other as long as the length of the stack, $s.length$, does not exceed $l.length$. If the width of the next piece π_j is different from $width(\pi_i)$ or $l.length$ is exceeded by $s.lenght$ then a new stack is started with π_j if $widht(\pi_j) \leq level.remWidth$, else a new level is started with π_j. This process is repeated until no pieces remain in the vector π.

Our version of the NFDH heuristic also allows stacked elements to extend a level's length if the total waste of the level decreases. Let us consider the example shown in Figure 2 with a level (the grey area represents the waste) and a piece to be arranged. The new piece can be placed into the current level if the area of the new piece is greater or equal than the area of the waste produced by the enlargement of the level (represented by a the rectangle with dotted lines in Figure 3).

2.3 Fitness Function

GAs are guided by the values computed by an objective function for each tentative solution until an optimum or an acceptable solution is found. In our problem, the objective is to minimize the strip length ($strip.length$) needed to build the layout corresponding to a given solution π. But two packing patterns can have the same length —so their fitness values will be equal— although, from the point of view of reusing the trim loss, one of them can be actually better because the trim loss in the last level (which still connects with the remainder of the strip) is greater than the one present in last level in the other layout. Therefore we are using the following, more accurate, fitness function:

$$F(\pi) = strip.length - \frac{1}{l.waste} \tag{1}$$

where $strip.length$ is the length of the packing pattern corresponding to the permutation π and $l.waste$ is the area of reusable trim loss in the last level of the packing pattern.

2.4 Genetic Operators

In this section we will describe the operators applied in all our algorithms. On the one hand, recombination of two tentative solutions will allow to intensify the search in the regions of the fitness landscape defined by the two parents (operator with arity 2). On the other hand, mutation will allow for the diversification of one solution (unary operator) that will hopefully allow to scape from local optima and also to enter new genetic diversity into the population since it will be gracefully exhausted by the algorithm as it converges during the search.

Several of this operators work in terms of the filling rates of the levels belonging to the parent solutions. This rate is calculated as follows for a given level l:

$$fr(l) = \sum_{i=1}^{n} \frac{width(\pi_i) \times length(\pi_i)}{W \times l.length} \tag{2}$$

where $\pi_1, ..., \pi_n$ are the pieces in l, $width(\pi_i)$ and $length(\pi_i)$ are the piece dimensions, and W and $l.length$ the level dimensions.

A new dedicated recombination operator, called *Best Inherited Level recombination* (BIL), take into consideration the pieces layout into their procedure. In section 4, several classical recombination operators are presented (PMX, OX, CX and EX), including a complete description of the the new BIL recombination operator for the 2SPP.

The idea behind the mutation operator used, named as *Best and Worst Stripe Exchange (BW_SE)*, is to change the location of the pieces so that the final trim loss is reduced. In BW_SE, the best and the worst level are always moved. The pieces of the best level (the one with highest filling rate) are allocated in the first positions of the new cutting pattern while the pieces of the worst level are assigned to the last positions. The middle positions are filled with the remaining pieces in the order they appeared in the original cutting pattern. In BW_SE, the movements can help to the involved levels or their neighbors to take pieces from neighboring levels improving their overall trim loss.

3 RECOMBINATION OPERATORS

In this section we will describe the five recombination operators studied. The first four operators have been proposed for permutation representation to solve the travelling salesman problem. These operators focus on combining order or adjacency information from the two parents. Operators such as Partial-Mapped Crossover (PMX), Order Crossover (OX), Cycle Crossover (CX) take into account the position and order of the pieces as opposed to edges (i.e., links between pieces). While the general idea behind the Edge Recombination (ER) is to preserve the linkage of a piece with other pieces. Finally, the fifth new operator, BIL, incorporates some problem-specific knowledge into their mechanism in order to improve the trim loss. We will now discuss all of them.

- *Partial Mapped Crossover (PMX).* This operator was proposed by Goldberg and Lingle [7]. It builds an offspring by choosing a subsequence of a cutting pattern form one parent and preserving the order and position of as many pieces as possible from the other parent. PMX can be viewed as an extension of two-point crossover for binary string to permutation representation. It uses a special repairing procedure to resolve the illegitimacy caused by the simple two-point crossover. Thus, in essence, PMX is a simple two-point crossover plus a repairing procedure.
- *Order Crossover (OX).* OX was proposed by Davis [5]. Since partial cuttings residing inside a solution seem to be the natural building blocks in this problem, our intention is that the recombination transmits these building blocks from parents to offspring. For this, two points are randomly selected from a parent, thus defining a crossover region. This region is transmitted directly to the offspring; meanwhile, the remaining positions are filled with the elements that do no belong to that region in the order that they appear in the second parent. It can be viewed as a kind of variation of PMX using a different repairing procedure.
- *Cycle Crossover (CX).* CX was proposed by Oliver et al. [20]. It takes some pieces (and their positions) from one parent which define a cycle according to the corresponding positions between parents. The remaining pieces are selected from the other parent.
- *Edge Recombination (EX).* EX was developed by Whitley et al. [26] and further enhanced in [24]. This operator transfers edges (relations between pieces) present in both parents. This is done with the help of an edge list created from both parents, which provides, for each piece i, all other pieces next to the piece i in at least one of the parents.

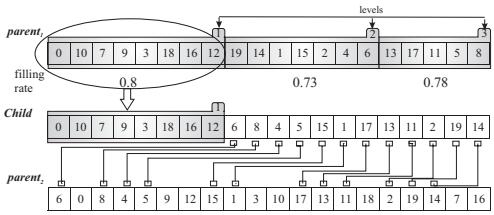

Figure 4: Example of BIL recombination.

- *Best Inherited Level Recombination* (BIL). This operator transmits the best levels of the parent to the child, i.e. those with the highest filling rate or, equivalently, with the least trim loss. In this way, the inherited levels can be kept or readjusted if one of them takes some pieces from the following level. This depends on how compacts the levels are: if they are very compact the levels should possibly be kept up and do not take pieces out of its neighboring level. The actual recombination works as follows. In the first step the filling rate of all levels from one parent, $parent_1$, is calculated. After that, a selection probability for each level l proportional to the filling rate is determined. A number k of levels are selected from $parent_1$ regarding proportional selection according to the filling rate. The pieces π_i belonging to the inherited levels are placed in the first positions of the child. Meanwhile, the remaining positions are filled with the pieces which do not belong to that levels in the order they appear in the other parent $parent_2$. Figure 4 gives an example for the level transfer in the course of a recombination operation and also the filling process of the remaining positions.

4 IMPLEMENTATION

Let us discuss in this section the actual implementation of the algorithms to ensure that this work is replicable in the future, a capital issue in research that is not always dealt with in the literature. As we said from the beginning, a sequential steady state GA will be here evaluated for a set of selected instances of the packing problem, in order to better know on their algorithmic effectiveness. The GA approach will be tested with different combinations of the presented recombinations (OX, PMX, BIL, EX and OX) and the mutation operator (BW_SE). The algorithm runs in MALLBA [1], a C++ software library fostering rapid prototyping of hybrid and parallel algorithms. All the algorithms were run until a maximum number of evaluations (2^{16}) is reached.

We considered here five classes of randomly generated problems with M equal to 100, 150, 200, 250 and 300 pieces. The instances were obtained by an own implementation of a data set generator following the ideas proposed in [25].

The steady state GA uses a binary tournament selection for each parent, and the new generated individual replace the worst individual in the population only if it is fitter. The population size for GA is set to 512 individuals. The initial population is randomly generated. The maximum number of evaluations is 2^{16}. The recombination operators are applied with a probability of 0.8, while the mutation probability was set to 0.1. Parameters (population size, stop criterium, probabilities, etc) were not chosen at random, but rather by an examination of values previously used with success (see [23] for example). Our machines have the following characteristics: Pentium 4 at 2.4 GHz and 512 MB RAM linked by Fast Ethernet. The operating system used is SuSE Linux with 2.4.19-4GB kernel version. See Table 2 for a summary of the parameters.

Table 2: Parameters used for the different GAs.

Parameter	GA
population size	512
recombination	PMX, OX, BIL, EX and CX
mutation	BW_SE
recombination probability	0.8
mutation probability	0.1
replacement strategy	$(\mu + 1)$
stop criterion	2^{16} evaluations

Table 3: Experimental results for the GA with all recombination operators.

Inst	**PMX**		**OX**		**BIL**		**EX**		**CX**	
	best	***avg*$\pm\sigma$**	***best***	***avg*$\pm\sigma$**	***best***	***avg*$\pm\sigma$**	***best***	***avg*$\pm\sigma$**	***best***	***avg*$\pm\sigma$**
100.00	265.00	284.04 ± 8.67	251.00	268.93 ± 9.31	**238.00**	249.39 ± 5.30	270.99	304.41 ± 14.90	314.00	338.71 ± 7.87
150.00	300.00	330.83 ± 13.49	283.00	301.05 ± 8.15	**240.00**	254.85 ± 6.79	310.00	351.46 ± 22.11	390.00	421.51 ± 10.80
200.00	300.00	326.52 ± 14.35	274.00	297.85 ± 8.98	**248.00**	257.78 ± 4.94	317.00	352.83 ± 18.43	386.00	411.11 ± 9.49
250.00	324.00	349.04 ± 13.70	303.00	314.09 ± 7.05	**245.00**	256.79 ± 5.23	332.00	386.09 ± 18.59	396.00	418.60 ± 9.25
300.00	326.00	363.54 ± 16.31	308.00	331.65 ± 10.00	**255.00**	266.52 ± 7.68	358.00	400.33 ± 23.93	426.00	455.15 ± 11.08
Mean	303.00	330.80	283.80	302.72	245.20	257.07	317.60	359.02	382.40	409.02

5 COMPUTATIONAL ANALYSIS

In this section we will summarize the results of using the proposed algorithm with its variants (recombination operators) on all the problem instances. For each algorithm we have performed 90 independent runs per instance using the parameter values summarized in Table 2. Our aim is to offer meaningful results from a statistical point of view. Also the evaluation considers two important issues for the whole search process: the capacity to generate new potentially promising individuals and the quickly progress in the surroundings of the best found solution. Hence, we considered the average fitness values for the first criterion and the entropy measure (as proposed in [8]) for the second one, which is computed as follows:

$$entropy = \frac{\sum_{i=1}^{M} \sum_{i=1}^{M} (\frac{n_{ij}}{\mu}) \ln(\frac{n_{ij}}{\mu})}{M \ln M} \tag{3}$$

where n_{ij} represents the number of times the piece i is set into the position j in the population of size μ. This function takes values in [0..1] and a value of 0 indicates that all the individuals in the population are identical.

Table 3 summarizes the results obtained for the GA when they apply all the considered recombination operators for each instance. The most relevant aspects measured in this comparison are the following ones: the best found feasible solutions (column *best*) and the average objective values of the best found feasible solutions along with their standard deviations (column $avg_{\pm\sigma}$). The last row in Table 3 shows average results over all the instances obtained by each algorithm just as a summary of the trends. The minimum *best* values are printed in bold.

The results clearly show that the GA applying the BIL operator outperform significantly the GA using any other recombination in terms of solution quality in all instances, and the difference is more evident as the instance dimension (regarding the number of pieces M) increases (see Figure 5 for more details). The reason for the outperforming is based on the improvement in the pieces layout obtained by the application of BIL, i.e. this operator is specific for this problem since it incorporates some knowledge of the problem inside their mechanism. Using the test of multiple comparisons of Bonferroni [3], we verified that the differences among the results are significant.

Figure 6 illustrates the behavior of the studied operators over all instances regarding average number of evaluations to reach the best value. In this Figure the X axis represents the

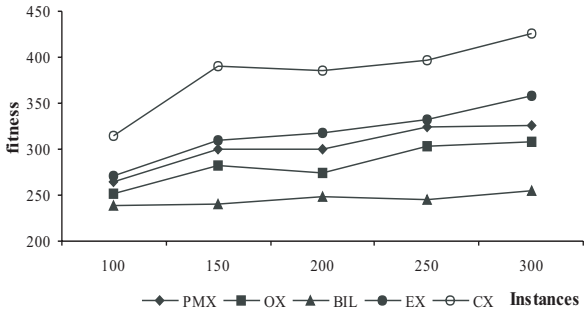

Figure 5: Best value for each algorithm and instance.

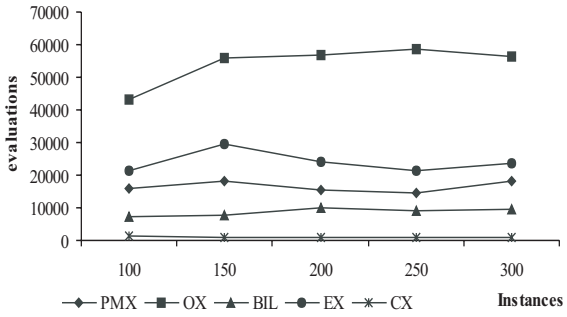

Figure 6: Mean number of evaluations to reach the best value for each algorithm and instance.

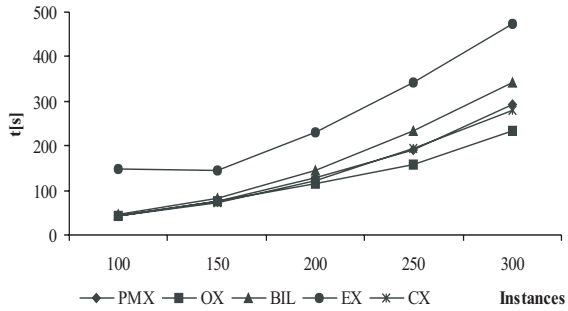

Figure 7: Mean spent time for each algorithm and instance.

different instances, while the Y axis indicates the average number of evaluations. CX reaches the best values using less average number of evaluations, but this algorithm presents the worst results regarding accuracy. A good performance is obtained for the algorithm applying the BIL operator and the difference in number of evaluations is significantly lower in comparison with GA plus OX. To confirm these results, we used the test of multiple comparisons of Bonferroni, which indicates that the difference among the algorithms are significant under this metric.

On the other hand, Figure 7 shows the time spent in the search (in seconds) for the algorithms applying the studied operators over all instances. The X axis represents the different instances, while the Y axis indicates the average time for each algorithm. The fastest approach is when applying OX, due to its simplistic mechanism to generate the offspring. Meanwhile the slowest one is the GA using EX; the reason is the complex procedure to find the linkage among pieces. Algorithms using BIL, OX and PMX spend very similar times in the search process; but a difference between BIL and the other two are observed as M increases. This is due to the search of the best levels to transmit and principally to the time incurred in the search of the pieces that not belong to the k levels which do not transmitted already. Using the test of multiple comparisons of Bonferroni, we verified that OX and CX present very similar execution times while the differences among the other recombination operators are significant.

In order to analyze the trade-off between exploration and exploitation, the behavior of the studied operators is illustrated over the instance with M=200 (similar results are obtained with the rest of the instances), in function of the evolution of average population fitness and entropy. Figure 8 presents the evolution of the population entropy (Y axis) with respect to the number of generations (X axis) while Figure 9 plots the average population fitness (Y axis). We can observe that the best trade-off between exploration and exploitation is obtained by the BIL operator, because it presents the lowest average population fitness together with high entropy values and a quick convergence.

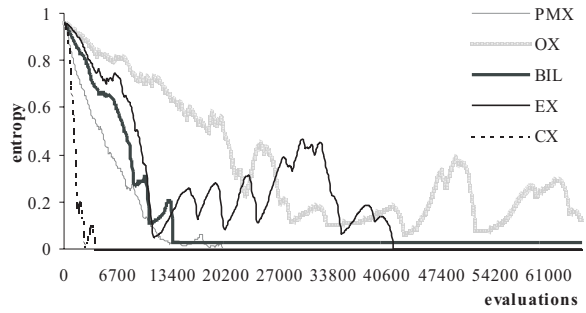

Figure 8: Population entropy for all the algorithms (instance with M=200).

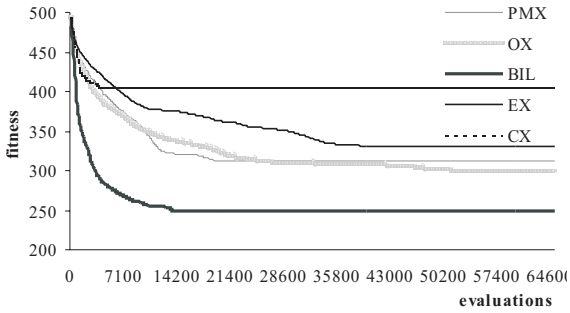

Figure 9: Average population fitness for all algorithms (instance with M=200).

To summarize the findings of this section, it is clear that the GA using the BIL operator, a problem-specific mechanism, computes very good average values regarding all the metrics analyzed in this work, and, moreover it presents the best trade-off between exploration and exploitation.

6 CONCLUSIONS

In this paper we intend to design better algorithms to solve the 2SPP. The selected research line is that of hybridization. For that we compare different recombination operators against the best inherited level recombination (BIL), included all of them in the basic search template of a GA. The study validates results from a statistical point of view, as well as analyzes the capacity of the recombination operators to generate new potentially promising individuals and the ability to keep a diversified population.

Our results show that our new recombination operator incorporating specific knowledge from the problem works properly. It provides a good sampling of the search space in all instances and also exhibits a satisfactory trade-off between exploration and exploitation of the search space.

We noticed that the used problem representation is maybe decoupling the search of the algorithms from the actual problem. In the future we will investigate non-permutation representations and a direct mapping to the final disposition of the pieces.

ACKNOWLEDGEMENTS

This work has been partially funded by the Spanish Ministry of Education and the European FEDER under contract TIN2005-08818-C04-01 (the OPLINK project, http://oplink.lcc.uma.es). We also acknowledge the Universidad Nacional de La Pampa, and the ANPCYT in Argentina from which we received continuous support.

REFERENCES

[1] E. Alba, J. Luna, L.M. Moreno, C. Pablos, J. Petit, A. Rojas, F. Xhafa, F. Almeida, M.J. Blesa, J. Cabeza, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, and C. León. *MALLBA: A Library of Skeletons for Combinatorial Optimisation*, volume 2400 of *LNCS*, pages 927–932. Springer, 2002.

[2] T. Bäck, D. Fogel, and Z. Michalewicz. *Handbook of evolutionary computation*. Oxford University Press, New York, 1997.

[3] C.E. Bonferroni. Teoria statistica delle classi e calcolo delle probabilitá. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62, 1936.

[4] A. Bortfeldt. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research (article in press)*, 2005.

[5] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[6] S.P. Fekete and J. Schepers. On more-dimensional packing III: Exact algorithm. Technical Report ZPR97-290, Mathematisches Institut, Universität zu Köln, available from the first author at Department of Mathematics, 1997.

[7] D.E. Goldberg and R. Lingle. Alleles, loci, and the tsp. *Grefenstette, J.J., (editor), Proceedings of the First International Conference on Genetic Algorithms*, pages 154–159, 1985.

[8] J.J. Grefenstette. *Genetic Algorithms and Simulated Annealing*, chapter Incorporating problem specific knowledge into genetic algorithms, pages 42–60. Morgan Kaufmann Publishers, 1987.

[9] E. Hopper. *Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods*. PhD thesis, University of Wales, Cardiff, U.K., 2000.

[10] E. Hopper and B. Turton. An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. *European Journal of Operational Research*, 128(1):4–57, 2000.

[11] E. Hopper and B. Turton. A review of the application of meta-heuristic algorithms to 2d strip packing problems. *Artificial Intelligence Review*, 16:257–300, 2001.

[12] S. Hwang, C. Kao, and J. Horng. On solving rectangle bin packing problems using genetic algorithms. *IEEE International Conference on Systems, Man, and Cybernetics - Humans, Information and Technology*, 2:1583–1590, 1994.

[13] S. Jakobs. On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, 88:165–181, 1996.

[14] B. Kroger. Guillotineable bin-packing: a genetic approach. *European Journal of Operational Research*, 84:645–661, 1995.

[15] D. Liu and H. Teng. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. *European Journal of Operational Research*, 112:413–420, 1999.

[16] A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: a survey. *European Journal of Operational Research*, 141:241–252, 2002.

[17] S. Martello, S. Monaci, and D. Vigo. An exact approach to the strip-packing problem. *Informs Journal on Computing*, 15:310–319, 2003.

[18] M. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, third revised edition, 1996.

[19] C.L. Mumford-Valenzuela, J. Vick, and P.Y. Wang. *Metaheuristics: Computer Decision-Making*, chapter Heuristics for large strip packing problems with guillotine patterns: An empirical study, pages 501–522. Kluwer Academic Publishers BV, 2003.

[20] I.M. Oliver, D.J. Smith, and J.R.C. Holland. A study of permutation crossover operators on the traveling salesman problem. *Grefenstette, J.J., (editor), Proceedings of the Second International Conference on Genetic Algorithms*, pages 224–230, 1987.

[21] J. Puchinger, G.R. Raidl, and G. Koller. *Solving a Real-World Glass Cutting Problem*, volume 3004 of *LNCS*, pages 162–173. Springer, 2004.

[22] C. Salto, J.M. Molina, and E. Alba. Sequential versus distributed evolutionary approaches for the two-dimensional guillotine cutting problem. *Proceedings of International Conference on Industrial Logistics (ICIL 2005)*, pages 291–300, 2005.

[23] C. Salto, J.M. Molina, and E. Alba. Analysis of distributed genetic algorithms for solving cutting problems. *to appear in International Transactions in Operational Research*, 2006.

[24] T. Starkweather, S. McDaniel, K. Mathias, C. Whitley, and D. Whitley. A comparison of genetic sequencing operators. *R. Belew and L. Booker (editors), Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 69–76, 1991.

[25] P.Y. Wang and C.L. Valenzuela. Data set generation for rectangular placement problems. *EJOR*, 134:378–391, 2001.

[26] D. Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and traveling salesmen: the genetic edge recombination operator. *J. Schaffer (editor), Proceedings of the Third International Conference on Genetic Algorithms*, pages 133–140, 1989.