Optimisation of a Multi-Objective Two-Dimensional Strip Packing Problem based on Evolutionary Algorithms

Jesica de Armas, Coromoto Leon, Gara Miranda, Carlos Segura

Optimisation of a Multi-Objective Two-Dimensional Strip Packing Problem based on Evolutionary Algorithms

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2008-IJPR-0588.R2
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	18-Dec-2008
Complete List of Authors:	de Armas, Jesica; Universidad de La Laguna, Estadistica, I.O. y Computacion Leon, Coromoto; Universidad de La Laguna, Estadistica, I.O. y Computacion Miranda, Gara; Universidad de La Laguna, Estadistica, I.O. y Computacion Segura, Carlos; Universidad de La Laguna, Estadistica, I.O. y Computacion
Keywords:	CUTTING STOCK PROBLEMS, PACKING PROBLEMS, EVOLUTIONARY ALGORITHMS, PARETO OPTIMIZATION
Keywords (user):	MULTI-OBJECTIVE OPTIMISATION , TWO-DIMENSIONAL STRIP PACKING

International Journal of Production Research
Vol. 00, No. 00, 18 December 2008, 1–17

ARTICLE

Optimisation of a Multi-Objective Two-Dimensional Strip Packing Problem based on Evolutionary Algorithms

Jesica de Armas, Coromoto León, Gara Miranda*, and Carlos Segura

Avda. Astrofísico Fco. Sánchez s/n. Dpto. Estadística, I. O. y Computación. Universidad de La Laguna. 38271, La Laguna, Tenerife, Spain
Email: (jdearmas, cleon, gmiranda, csegura)@ull.es

(*v3.0 released December 2008*)

This paper considers a real-world Two-Dimensional Strip Packing Problem involving specific machinery constraints and actual cutting production industry requirements. To suit the problem to a wider range of machinery characteristics, the design objective contemplates the minimisation of material length and the total number of cuts for guillotinable-type patterns. The number of cuts required for the cutting process is crucial for the life of the industrial machines and is an important aspect in determining the cost and efficiency of the cutting operation. In this paper we propose the application of evolutionary algorithms to address the multi-objective problem, for which numerous approaches to its single-objective formulation exist but for which multi-objective approaches are almost non-existent. The multi-objective evolutionary algorithms applied provide a set of solutions offering a range of trade-offs between the two objectives from which clients can choose according to their needs. At the same time, by considering both the length and number of cuts, they derive solutions with wastage levels similar to most previous approximations which just seek to optimise the overall length.

Keywords: Two-Dimensional Strip Packing; Multi-Objective Optimisation; Evolutionary Algorithms.

1. Introduction

Cutting and packing problems arise in many industrial applications where it is necessary to arrange a set of required pieces or items on a stock sheet or raw material. It is essential to produce good quality arrangements which allow for raw material utilisation and, therefore, wastage to be minimised. Cutting and packing problems appear in many forms [Dyckhoff (1990), Dowsland and Dowsland (1992)] and involve different dimensions, types of pieces (irregular or regular), types of cuts (orthogonal, guillotine, non-guillotine), piece rotation (fixed, 90°) or final goal. Most variants of cutting and packing problems have been widely studied [Sweeney and Paternoster (1992), Lodi *et al.* (2002)] and many approaches have been proposed. Among the existing approximations, three kind of techniques can be distinguished: exact, heuristics, and metaheuristics. Initially, a wide variety of exact methods were devised [Bekrar *et al.* (2007)]. However, such approaches cannot accommodate large and actual problem instances. For this reason, heuristic strategies were formulated in order to obtain good quality - although not necessarily optimal - solutions in an acceptable computational time [Dagli and Tatoglu (1987), Mumford-Valenzuela *et al.* (2004)]. As a more general and sophisticated method, different types of hybrid

*Corresponding author. Email: gmiranda@ull.es

ISSN: 1478-6451 print/ISSN 1478-646X online

DOI: 10.1080/1478645YYxxxxxxx
http://www.informaworld.com

algorithms and metaheuristics have been considered [Hopper and Turton (2001), Bortfeldt (2006), Garrido and Riff (2007), Padmanaban and Prabhaharan (2008)]. Some metaheuristics cited above generate a number of different solutions or input sequences that are usually interpreted by placement heuristics. These algorithms are guided through the problem search space by previous attempts and involve a set of parameters that must be fine-tuned so as to improve the resulting performance.

This paper focuses on the so-called *Two-Dimensional Strip Packing Problem* (2DSP) where a set of specified pieces must be obtained from a large stock sheet with fixed width W and unlimited length. Each specified rectangular piece i has fixed dimensions (l_i, w_i), although it can be rotated 90 degrees, considering also the dimensions (w_i, l_i). All specified pieces must be orthogonally arranged on the material so as not to overlap and so that only vertical or horizontal builds (Figure 1) of pieces are generated. This last constraint always yields the possibility of producing the final patterns through guillotine cuts, i.e. the pieces have to be cut with their edges parallel to the edges of the stock sheet going from one border straight to the opposite side. The production of non-guillotine cuts may require a more complex machinery operation. For this reason, in order to give support to a wider range of industrial cutting machines, in the proposed approach only guillotinable patterns are built so that all resulting solutions can be cut in both guillotine and non-guillotine modes.

The most common goal in the 2DSP consists of minimising the necessary stock sheet length required to cut the whole set of specified pieces. However, in some 2D real-world cutting and packing applications, other optimisation criteria may be taken under consideration. In some industrial fields, the raw material is either very cheap or can be easily recycled. In such cases, a more important criterion for the pattern generation may be the speed at which the pieces can be obtained, thus maximising the usage of the cutting equipment [Cowton and Wirth (1993)]. The cutting speed is specifically limited by the features of the available machinery but, in general, it is determined by the number of cuts involved in the cutting pattern. Moreover, the number of cuts required for the cutting process is also crucial for the life of the industrial machines. Since the number of cuts is an important aspect in determining the cost and efficiency of the cutting operation, a comprehensive optimisation methodology should take also this criterion into consideration.

Therefore, in this study, the number of cuts is taken as a second design objective. This way, the problem can be posed as a multi-objective optimisation problem for optimising the layout of rectangular parts so as to minimise both the sheet length (trim loss) and the number of cuts necessary to achieve the final specified pieces. Many proposals appear in the literature considering the single-objective formulation of the problem, but only a few address real-world multi-objective constraints [Song *et al.* (2006), Tiwari and Chakraborti (2006), Illich *et al.* (2007)]. In Song *et al.* (2006) several optimisation criteria are analysed for the strip packing problem. The application of multi-objective evolutionary algorithms is described in Tiwari and Chakraborti (2006) and Illich *et al.* (2007) for solving the two-objective optimisation problem involving the minimisation of the material length and the number of cuts. The first approach always allows for guillotinable cuts, while the second approach only allows for non-guillotine cuts.

The remaining content of the article is structured as follows: Section 2 briefly introduces multi-objective optimisation problems and outlines the most common approaches for their solution. A description of the problem implementation details and its specific features for developing an evolutionary approach are presented in Section 3. The computational results obtained with some of the most widely used evolutionary algorithms and further comparisons with single-objective approxima-

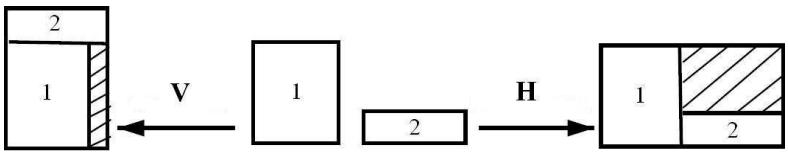

Figure 1. Vertical and horizontal builds

tions are shown in Section 4. Finally, the paper closes with the conclusions and some future lines of work.

2. Multi-Objective Optimisation

Multi-objective or multi-criteria optimisation problems (MOPs) arise in most real-world disciplines [Dimopoulos (2006), Ye and Zhou (2006), Nearchou (2007)]. The multiple objectives are usually conflicting or competing, but must be simultaneously optimised. In practise, it is often impossible to find a single optimum that dominates all other solutions Therefore, a solution optimising every single objective might not exist. MOPs can be formally described as:

$$Optimise\ f(x) = (f_1(x), f_2(x), ..., f_k(x)) \quad subject\ to\ x \in X \tag{1}$$

where $f(x)$ is the objective vector, $f_i(x)$ is the i-th objective to be optimised, x is the decision vector and X is the feasible region in the decision space. While in single-objective optimisation the optimal solution is usually clearly defined, this does not hold for MOPs. Instead of a single optimum, there is rather a set or front of alternative trade-offs, known as *Pareto-optimal* front, constituted by the non-dominated solutions y:

$$\nexists z / \ \forall i \in 1..k \quad f_i(z) \geq f_i(y), \exists i \in 1..k \quad f_i(z) > f_i(y) \tag{2}$$

These solutions are optimal in the sense that no other solutions in the search space are superior to them when all objectives are considered. That is, a solution y is said to be non-dominated if there does not exist a solution z that performs better in at least one objective and at least as well as y in the rest. The final aim when dealing with MOPs is to obtain a non-dominated solution set which, in the best case, will coincide with the Pareto-optimal front. From the resulting final solution set, a human decision maker will be able to select a suitable compromise solution.

2.1 *Multi-Objective Approaches*

Since exact approaches are practically unaffordable for most MOPs, a wide variety of approximated algorithms have been designed. Two common approaches for simplifying the MOPs solution are: convert the original problem into a single-objective one by combining or aggregating the multiple objectives into a single function; or, translate some of the objectives into constraints. Optimising a combination of the objectives has the advantage of producing a single compromise solution - which can be achieved by the application of classical single-objective optimisation strategies - requiring no further interaction with the decision maker. However, the application of such an approach requires a prior knowledge of the problem that is not always available. Moreover, these techniques lose in solution diversity and may be sensitive to the shape of the Pareto-optimal front, e.g. if the optimal solution cannot be

accepted, either due to the function used excluding aspects of the problem which were unknown prior to optimisation, or due to an inappropriate setting of the coefficients of the combining function, new runs of the optimiser may be required before a suitable solution is found [Fonseca and Fleming (1995), Zitzler (1999)].

Therefore, a MOPs solution calls for alternative approaches. Usually, a more appropriate approximation involves the application of techniques that can specifically deal with multiple objectives and MOPs intrinsic complexity (very large search spaces, uncertainty, noise, disjoint Pareto curves, etc.). *Evolutionary algorithms* (EAs) have shown great promise for calculating solutions to large and difficult optimisation problems and have been successfully used across a wide variety of real-world applications [Eiben (1998)]. In fact, when applied to MOPs, EAs seem to perform better than other blind search strategies. The use of EAs to solve problems of this special nature has been motivated mainly because they are able to capture multiple Pareto-optimal solutions in a single simulation run - which is possible thanks to their population-based feature - and to exploit similarities of solutions by recombination.

EAs that are specifically designed to deal with multiple objective functions are known as *multi-objective evolutionary algorithms* (MOEAs) [Coello *et al.* (2007)]. When designing MOEAs two major problems must be addressed [Zitzler *et al.* (2000)]: how to accomplish fitness assignment and selection in order to guide the search towards the Pareto-optimal set, and how to maintain a diverse population in order to prevent premature convergence and achieve a well distributed trade-off front. Many alternatives have been proposed in an attempt to adhere to such design goals [Coello (1999)]: VEGA [Schaffer (1985)], NPGA [Horn and et al. (1994)], NSGA [Srinivas and Deb (1994), Deb *et al.* (2002)], SPEA [Zitzler and Thiele (1998a), Zitzler *et al.* (2002)], IBEA [Zitzler and Künzli (2004)], etc.

3. Problem Implementation

Evolutionary approaches have been successfully applied to the single-objective 2DSP formulation [Ono and Ikeda (1998), Mumford-Valenzuela *et al.* (2004), Bortfeldt (2006)], achieving competitive results for most test problems even when compared to some heuristics specifically designed to address the 2DSP. Moreover, MOEAs have demonstrated, in general, to behave better than other blind search strategies in the optimisation of real-world MOPs. In fact, to our knowledge, the only two approaches for the multi-objective 2DSP are both based on MOEAs. So, in this work, the problem solution also lies in the application of MOEAs.

In order to simplify the 2DSP solution, a C++ framework that provides some of the best known MOEAs has been used [León *et al.* (2007)]. The plug-in architecture of the framework allows final users to incorporate their own problems and evolutionary algorithms. Although some specific EA-based frameworks have been proposed for solving MOPs [Gagné and Parizeau (2006), Liefooghe *et al.* (2007)], not many support parallel schemes and even fewer provide an easy customisation of the parallel models. As a novelty, the selected framework integrates a new parallel island-based model - called *team algorithm* - that allows for the dynamic cooperation and self-adaption of different MOEAs in the solution of a given problem [León *et al.* (2008)]. Although the tool allows for ample customisation of the execution models - including several parallel schemes - in this case its default sequential behaviour was sufficient to successfully deal with the problem.

To obtain a solution to the 2DSP using the tool, the user must select the set of MOEAs to be employed together with their set of parameters. Moreover, it is also necessary to define some specific features related to the problem, i.e. gene

codification for the representation of the individuals, the functions involved in the evaluation of the objectives, and also the genetic operators to be applied during the evolution process.

3.1 *Optimisation algorithms*

In particular, from the MOEAs provided by the framework, the ones tested for the solution of the 2DSP are briefly described next:

- NSGA-II [Deb *et al.* (2002)] is a non-dominated sorting based MOEA. Two of the most important characteristics that differentiate NSGA-II from NSGA and other non-dominated sorting based approaches are: a fast non-dominated sorting approach with reduced computational complexity and a selection operator which combines previous populations with new generated offspring, ensuring elitism in the approach. The algorithm works as follows: the two populations are sorted according to their rank, and the best solutions are chosen to create a new population. In the case of having to select some individuals with the same rank, a density estimation based on measuring the crowding distance to the surrounding individuals belonging to the same rank is used to yield the most promising solutions.
- SPEA2 [Zitzler *et al.* (2002)] uses a population and an archive. It assigns to each individual a fitness value that is the sum of its strength raw fitness plus a density estimation. In each generation the non-dominated individuals of both the original population and the archive are used to update the archive. If the number of non-dominated individuals is greater than the population size, a truncation operator based on calculating the distances to the k-th nearest neighbour is used. This entire procedure is known as *environmental selection*. Then, the algorithm applies the selection, crossover, and mutation operators to members of the archive in order to create a new population of offspring which becomes the population of the next generation.
- IBEA [Zitzler and Künzli (2004)] allows the optimisation goal to be defined in terms of a performance measure or *quality indicator*. This measure is used directly for fitness calculation. The algorithm allows the use of different binary quality indicators. In this work the binary multiplicative ϵ-indicator [Zitzler *et al.* (2003)] was used. There exist two versions of IBEA, one basic and a more robust version known as adaptive. In the adaptive version, which is the one applied here, objectives values are normalised and the indicator values are adaptively scaled.

3.2 *Gene codification*

In this work a postfix notation is used to represent the candidate solutions [Ono and Ikeda (1998), Tiwari and Chakraborti (2006)]. The operands are the identifiers of the pieces, while the operators are 'V' and 'H' (Figure 1). The operator 'H' concatenates its two operands horizontally, while the 'V' operator concatenates them vertically unless the problem width constraint is violated, in which such case it behaves as the 'H' operator. Note that the 'H' operator can always be applied without violating the problem constraints because the stock sheet has unlimited length. At the end of the gene, a bit string equal in size to the number of pieces is attached. It determines whether each piece must be rotated. In order to constitute a valid gene, each piece must appear once. Moreover, for any operator, if the number

of pieces to its left is n_p and the number of operators to its left plus itself is n_o, then the following condition must hold:

$$1 \leq n_o \leq n_p - 1 \tag{3}$$

Using such a representation, the gene size remains constant and no parentheses are required to uniquely represent a solution (Figure 2). Note also that the gene representation defines cutting patterns built by means of vertical and horizontal constructions, thus always providing a packing solution that can be cut in the guillotine mode and even, if the appropriate machinery is available, using a non-guillotine process.

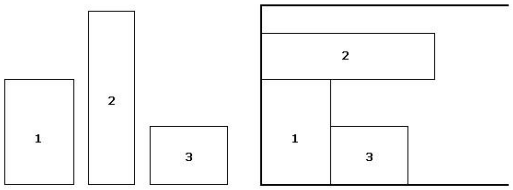

Figure 2. Layout in the stock sheet for the gene '1 3 H 2 V 010'

3.3 *Evaluation of the objectives*

As mentioned before, every gene represents a certain arrangement of pieces on the stock sheet. The chosen codification gives information on how pieces must be combined or placed on the raw material. Based on such information, both optimisation objectives considered, i.e. the overall length and the number of cuts necessary, can be evaluated. For this purpose, the methods applied here are based on the usage of stacks and the postfix notation which represents the gene [Ono and Ikeda (1998)].

In evaluating the second objective - the number of cuts necessary - two different types of evaluations must be considered since the gene representation used allows for both guillotine and non-guillotine cutting processes. Whenever guillotine cutting is not necessary, the number of cuts required to accomplish the job can be significantly reduced if already cut parts are laid undisturbed on the stock sheet and combined cuts are made for the remaining aligned edges. Placing a guillotine restriction on the cutting process alters the number of cuts involved in obtaining a given pattern set, thus yielding a different number of cuts for the same arrangement of pieces when considering guillotine or non-guillotine situations. Figure 3 shows how the same arrangement of pieces involves a different number of required cuts when guillotine or non-guillotine methods are applied.

In the case of guillotine cutting where the cut has to be made from edge to edge, an iterative method is applied to calculate the number of cuts involved. The gene is traversed from left to right, interpreting every element and creating the indicated constructions, thus calculating the partial widths and lengths. For each implied vertical or horizontal combination of pieces, at least one cut is necessary. If the combined rectangles do not match in length (for vertical builds) or in width (for horizontal builds), an extra cut is required for such a construction. At the end of the process the complete final pattern is obtained. In this case, the value of the first objective - required overall length - is immediately given by the length of the resulting final pattern. This procedure is depicted in Algorithm **1**. Note that the algorithm incorporates the corresponding pseudo-code to ensure the evaluation of genes always satisfies the stock sheet width constraint.

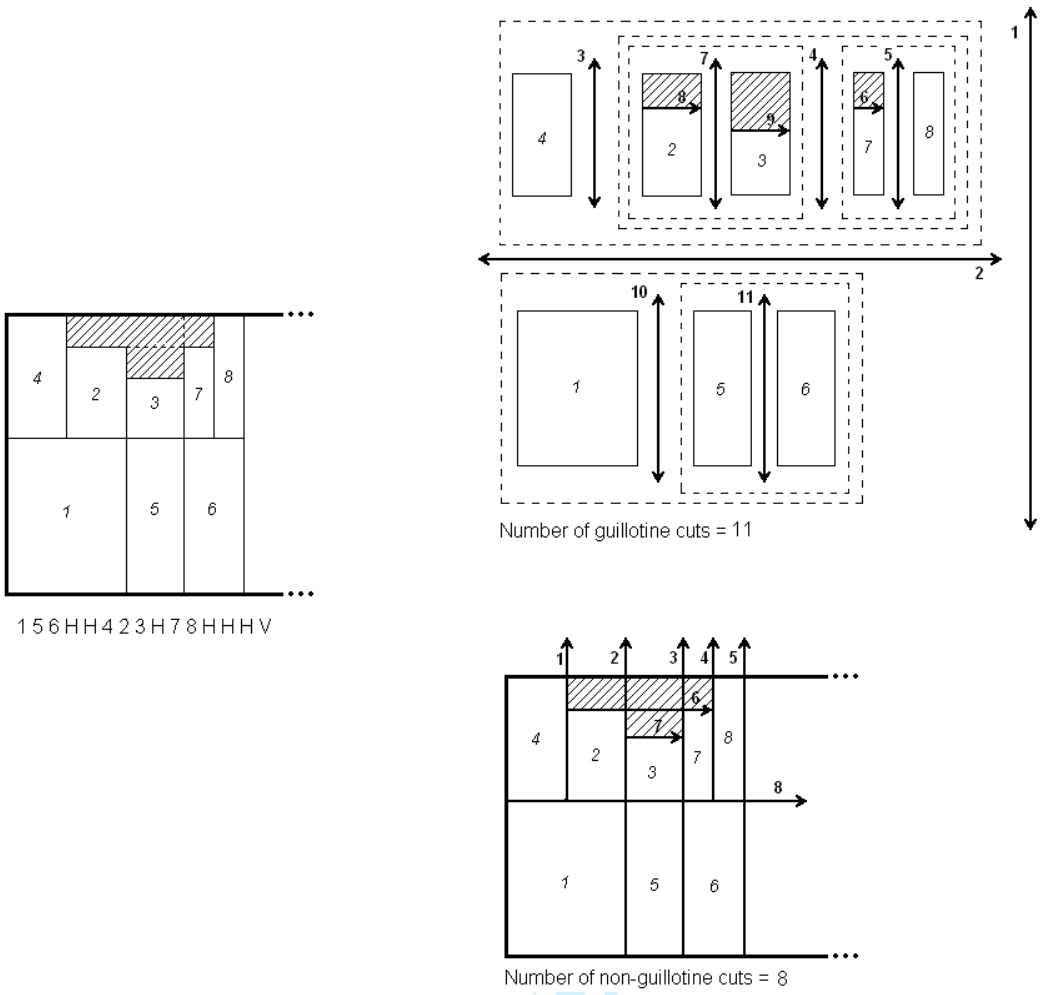

Figure 3. Comparison of guillotine and non-guillotine cutting processes

For the evaluation of the objectives in the non-guillotine mode, an initial analysis of the gene is needed in order to calculate the bottom-left (x_1, y_1) and top-right (x_2, y_2) coordinates of each item when placed in the stock sheet. The procedure starts with the final rectangle or pattern composition, which is then split into its two constituent parts. Then, the same method is recursively invoked until all the individual items are separated and their coordinates are calculated. As in the guillotine case, the required stock sheet length can be obtained from the dimensions of the final rectangle. For evaluating the number of cuts, it is assumed that all the pieces are laid out with their edges orthogonal to the stock sheet edges, with no overlap between them or the stock sheet edges. In this situation, the maximum number of cuts required in a cutting pattern composed by n pieces is $4n$, since each piece has four edges. Each edge of every piece is then checked to see if it lies on one of the stock sheet edges. If it does, such a cut is not required, thus the total number of cuts can be decreased by one. Next, the remaining edges which are not on the edges of the stock sheet are checked with the edges of all the other rectangles in order to find the alignments. For every pair of checked edges, only two situations are possible: the two rectangles touch each other (either at the corners or through their edges) or they do not touch at all. If they touch each other, then only one cut is required to obtain the cutting of both edges, thus decreasing the number of cuts by one. If the edges do not touch each other, then it is necessary to find the line joining the two edges and whether any other rectangle crosses that

Algorithm 1 Pseudo-code for the evaluation of objectives: Guillotine mode

```
numCuts ← 1;
for (i = 0 to sizeOfGeneVbles) do
  if (isOperator(gene(i)) == false) then
    stack.push(gene(i));
  else
    numCuts ← 1 + numCuts;
    pieceA ← stack.pop();
    pieceB ← stack.pop();
    if (gene(i) == H) then
      length(newPiece) ← length(pieceA) + length(pieceB);
      width(newPiece) ← max(width(pieceA), width(pieceB));
      if (width(pieceA) != width(pieceB)) then
        numCuts ← numCuts + 1;
      end if
    else
      length(newPiece) ← max(length(pieceA), length(pieceB));
      width(newPiece) ← width(pieceA) + width(pieceB);
      if (width(newPiece) ≥ width(stockSheet)) then
        gene(i) ← H;
        undo changes made in iteration i;
        repeat the iteration i;
      end if
      if (length(pieceA) != length(pieceB)) then
        numCuts ← numCuts + 1;
      end if
    end if
    stack.push(newPiece);
  end if
end for
finalPiece ← stack.pop();
totalLength ← length(finalPiece);
if (width(finalPiece) != width(stockSheet)) then
  totalCuts ← numCuts + 1;
else
  totalCuts ← numCuts;
end if
```

line. If no rectangle crosses such a line, a single cut suffices for both edges, again reducing the number of cuts by one. Note that for the same edge, the number of cuts can be reduced only once and also that for a given alignment of edges at least one cut should be computed. Algorithm **2** briefly depicts the evaluation method for the non-guillotine cutting mode. To simplify both pseudo-codes, considerations on the orientation of the pieces have been omitted.

3.4 *Genetic operators*

In general, a genetic algorithm maintains a population of candidate solutions for the problem, a population which is forced to evolve through the iterative application of a set of genetic operators. The most common genetic operators are: selection, crossover (also known as recombination) and mutation. Selection operators replicate the most successful solutions found in a population. Crossover mechanisms decompose two distinct solutions and then randomly mix their parts to form new solutions. A mutation randomly perturbs a candidate solution. During the evolution process, MOEAs, like other evolutionary strategies, apply genetic operators to the individuals of the current population. Selection operators allow the quality of the solutions in future generations to be preserved while crossover and mutation

Algorithm 2 Pseudo-code for the evaluation of objectives: Non-Guillotine mode

```
build finalPiece;
totalLength ← length(finalPiece);
calculate x1, y1, x2, y2 coordinates of finalPiece;
calculate x1, y1, x2, y2 coordinates of each piece inside finalPiece;
totalCuts ← 4 * numPieces;
for (i = 0 to numPieces) do
   numAxes ← edges of piece i that fit in lower, upper or left edges of stock sheet;
   totalCuts ← totalCuts - numAxes;
end for
for (each pair of pieces) do
   if (a pair of horizontal or vertical edges are aligned) then
      if (pieces do not touch each other) then
         if (there is another piece between both) then
            if (piece between does not cross the aligned edges) then
               totalCuts ← totalCuts - 1;
            end if
         else
            totalCuts ← totalCuts - 1;
         end if
      else
         totalCuts ← totalCuts - 1;
      end if
   end if
end for
```

operators generate the new individuals or offspring, thus maintaining diversity in the solutions. Although the selection is mainly defined by the particular MOEA to be used, the crossover and mutation depend on the specific problem to be solved, e.g. in the solution of the 2DSP the genetic operators must generate new individuals, ensuring the condition depicted in equation 3 is always satisfied.

The applied mutation [Ono and Ikeda (1998), Tiwari and Chakraborti (2006)] operates as follows. First, two gene elements, p_1 and p_2, are randomly selected. Both elements represent piece numbers or operators and p_1 is closer to the left of the gene. If both are piece numbers or operators, or p_1 is an operator and p_2 is a piece, they are swapped. If p_1 is a piece number and p_2 is an operator, they are swapped only when, after performing the swap, equation 3 still holds for any operator. Then, a gene operator is randomly chosen and flipped based upon the mutation probability. Finally, an orientation is also randomly chosen and flipped from '0' to '1' or vice versa depending again on the mutation probability.

For the crossover, the *Partially Mapped Crossover* [Goldberg and Lingle (1985)] has been selected, since its suitable behaviour has been demonstrated when applied to the 2DSP [Ono and Ikeda (1998), Tiwari and Chakraborti (2006)]. As in the case of the mutation operator, the crossover also ensures that only new valid genes are generated. The technique is based on the recombination of two gene chains where only the information of the pieces is taken into account, i.e. the operators and orientation bits are ignored for the application of this operator. Considering this type of chain, first, two crossing points inside each of the given parents are randomly chosen. Then, the segments of the parents inside such cross points are swapped in order to generate the offspring. The remaining chains in the offspring are obtained by mapping between the two parents. If a gene value outside the swapped segment is not contained in the swapped segment, it remains the same, but if it is already contained, it must be replaced by a value contained in the original segment of the gene but not contained in the new segment under consideration.

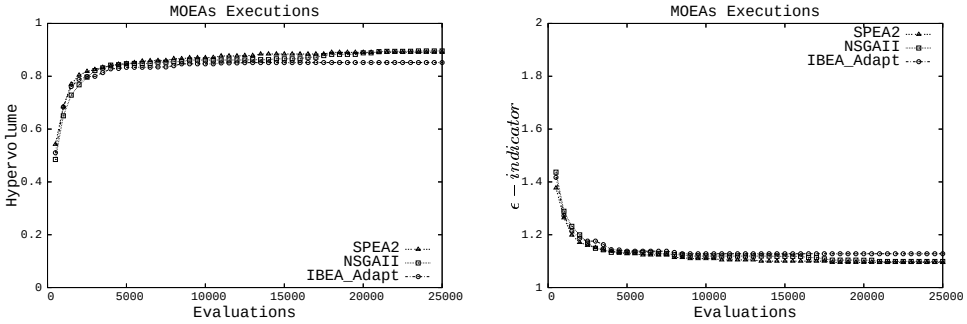

Figure 4. Metrics for Test Problem 4

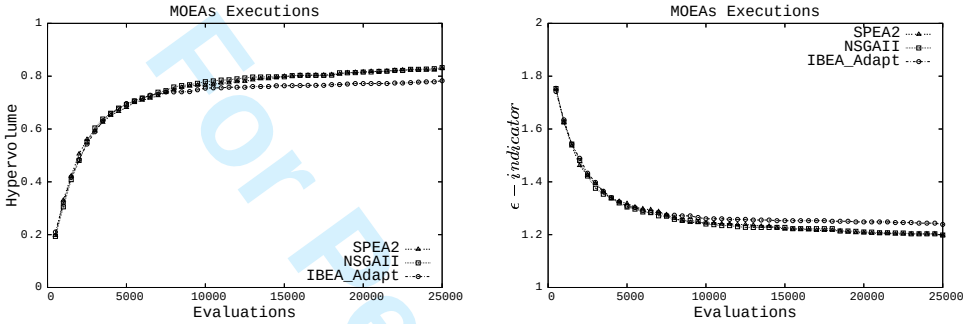

Figure 5. Metrics for Test Problem 5

4. Computational Results

The model was tested on a dedicated 20-node Debian GNU/Linux cluster with 1Gb RAM and two Intel® Xeon™ 2.66 GHz processors. The C++ compiler used was gcc 4.1.3. For the execution of the MOEAs, the crossover and mutation probabilities were set to $p_c = 0.7$ and $p_m = 0.3$, respectively, whereas the population size was set to $p_s = 50$. The values for this set of parameters were chosen as the result of a previous analysis in which the corresponding algorithms were tested and fine-tuned.

4.1 *Multi-objective optimisation*

The first experiment addresses the analysis of different MOEAs when dealing with the 2DSP. For the study, three of the most known MOEAs were considered: NSGA-II, SPEA2 and the adaptive version of IBEA. These algorithms are currently available in the multi-objective optimisation framework used [León *et al.* (2007)]. With the above specification of the gene representation and the implementation of the evaluation functions and the genetic operators, sequential executions for three such MOEAs were immediately available by using the framework. Although the tool provides several customisable parallel execution models, the analysis here does not focus on the behaviour of such parallel schemes. The study analyses the behaviour of multi-objective evolutionary approaches when dealing with a real-world MOP. In particular, for this initial study, a guillotine cutting process was assumed.

The multi-objective computational study presented in this work is based on evolutionary algorithms whose aim consists of improving the only other evolutionary multi-objective related work [Tiwari and Chakraborti (2006)]. In this work only five different test problems are defined. The first three problems are irrelevant because they are extremely simple and can be immediately and easily solved. That is why only the two larger problem instances proposed in Tiwari and Chakraborti (2006) were used for this experimental evaluation. Such larger instances are labelled as *test problem 4* and *test problem 5*. Test problem 4 considers a stock sheet

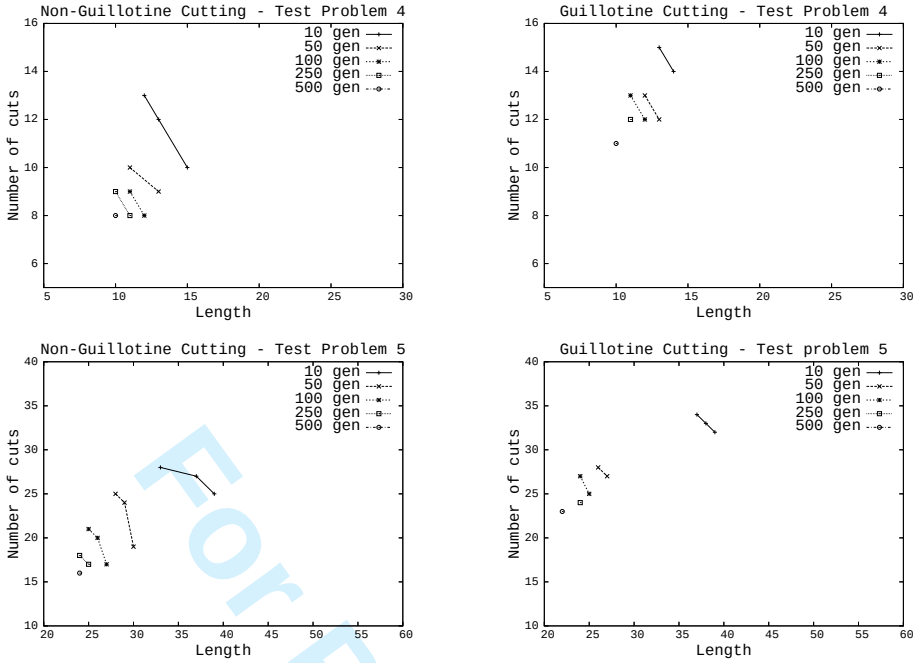

Figure 6. Evolution of Pareto fronts for an average execution

10 units wide and 10 different rectangles of various dimensions. Test problem 5 is composed of 20 pieces of different dimensions and uses a stock sheet 10 units wide. The three MOEAs were executed with both test problems number 4 and 5. Each type of execution was repeated thirty times and the average values considered.

In order to compare the results obtained for each problem with the three approaches considered, two different metrics were applied: hypervolume [Zitzler and Thiele (1998b)], and binary multiplicative ϵ-indicator [Zitzler *et al.* (2003)]. These metrics combine the analysis of convergence and diversity of solutions [Knowles *et al.* (2006)]. The binary multiplicative ϵ-indicator uses as reference front the union of the solutions obtained by all the experiments.

Figures 4 and 5 show the hypervolume and ϵ-indicator results for test problems 4 and 5, respectively. The metric values are represented from the beginning of the evolution process until 25000 individual evaluations (500 generations) are performed. The analysis of results for both metrics lead to similar conclusions. NSGA-II and SPEA2 results are very similar and slightly improve on those obtained by the IBEA approximation. Actually, NSGA-II appears to exhibit better behaviour, as noted in Tiwari and Chakraborti (2006). Moreover, its results remain more uniform between executions and it involves less computational effort than SPEA2 or IBEA.

Since NSGA-II seems to be the best MOEA from among those tested, its behaviour was analysed in depth. To this end, both non-guillotine and guillotine cutting processes were considered for test problems 4 and 5. Figure 6 shows the evolution of the fronts, from 10 generations (500 evaluations) up to 500 generations (25000 evaluations), for an average NSGA-II execution. Notice that test problem 4 is much simpler than test problem 5, and thus it usually converges faster and involves smaller Pareto fronts. For both problems, the final non-dominated solutions are achieved before 500 generations, thus demonstrating the good convergence of the evolutionary approach. Moreover, the results also demonstrate that the application of guillotine cuts requires a greater number of cuts than those needed for a non-guillotinable scenario.

Figure 7 shows the best Pareto fronts obtained by NSGA-II also for test problem

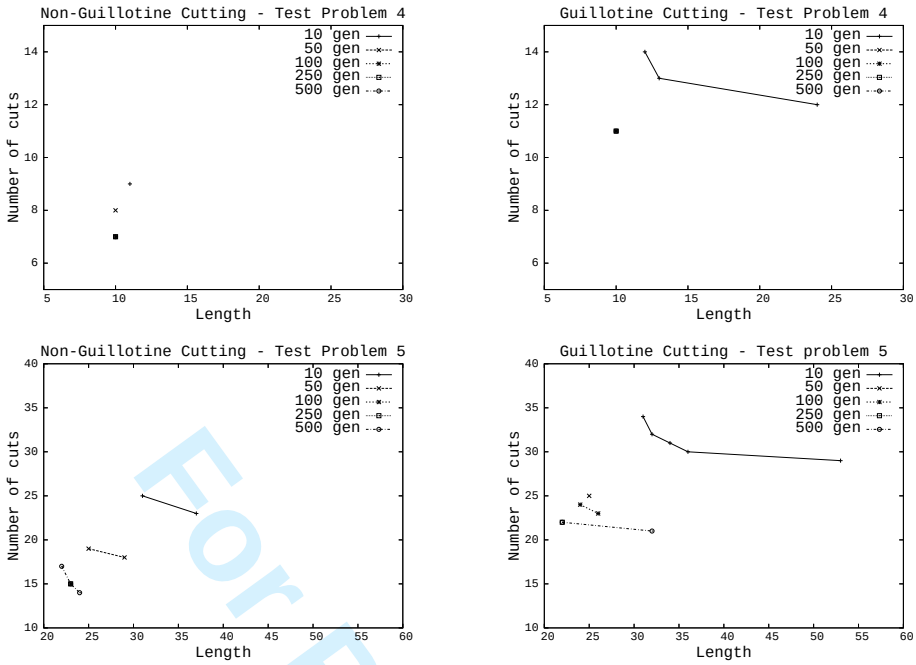

Figure 7. Evolution of Pareto fronts for the best execution

4 and 5. In this case, the Pareto fronts are smaller than those obtained in an average execution. The solutions of the average fronts are improved, although the difference is not very noticeable. As a layout example, Figure 8 shows the pattern distributions for the two solutions in the best Pareto front obtained for test problem 5 when applying guillotine-type cuts.

In order to validate the quality of the approaches proposed here, previous results were compared with those presented in Tiwari and Chakraborti (2006), which is - to our knowledge - the only other existing multi-objective approach for the guillotinable 2DSP. In the current work, the gene representation and the genetic operators applied are equivalent to those used in Tiwari and Chakraborti (2006). The main differences between the two approaches lies in the algorithms applied. In our case,

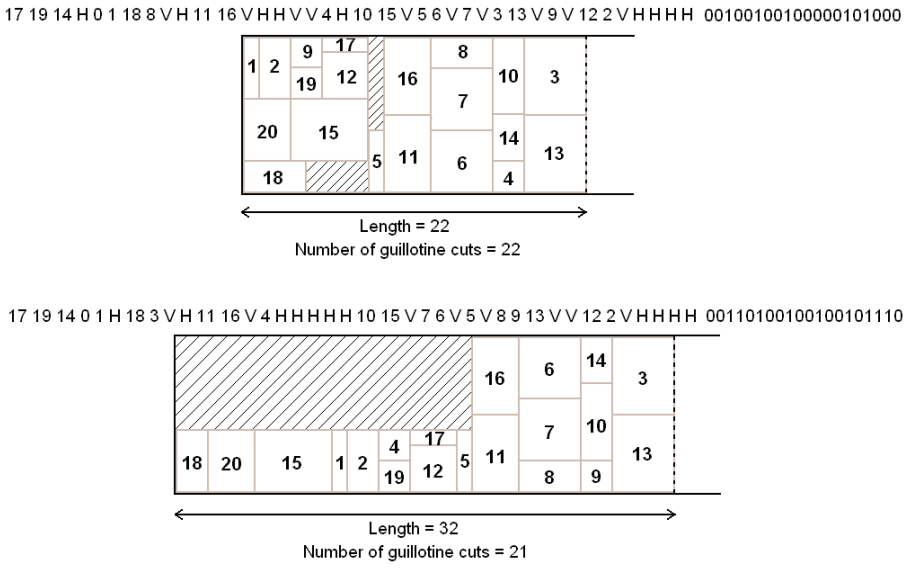

Figure 8. Guillotine layouts of the best achieved Pareto front for test problem 5

Table 1. Solutions for the existing multi-objective guillotinable approaches of the 2DSP.

	Non-Guillotine Cutting				Guillotine Cutting			
	Test 4		*Test 5*		*Test 4*		*Test 5*	
	Length	Cuts	Length	Cuts	Length	Cuts	Length	Cuts
Tiwari and Chakraborti (2006)	12	10	26	28	12	13	26	34
	14	9	27	25	14	12	27	30
			28	23			28	29
			29	22			29	28
			30	21			38	27
							46	26
							63	25
NSGA-II	10	7	22	17	10	11	22	22
			23	15			32	21
			24	14				

[a]The best values obtained are shown for each approach considered.

[b]Tiwari and Chakraborti (2006) approach involves a maximum of 100000 generations.

[c]The approach proposed here involves a maximum of 500 generations.

we used three different MOEAs provided by a multi-objective optimisation tool. In the work of Tiwari and Chakraborti (2006) the optimisation strategy employed is a modification of NSGA-II, although no further details of the algorithm or the parameter settings are given in the corresponding work.

Table 1 is included to allow for a better comparison between the only two multi-objective approaches for the guillotinable 2DSP. In the table, the best solutions for both approaches - Tiwari and Chakraborti (2006) and the one proposed here - are shown. Results are specified for the two possible cutting processes, non-guillotine and guillotine. The solutions obtained are represented as an overall length and an associated number of required cuts. The best solutions in Tiwari and Chakraborti (2006) are obtained after 100000 generations, while the best solutions achieved here require fewer than 500 generations. Although the previous approach provides larger Pareto fronts, the solutions obtained by the NSGA-II implementation proposed here are clearly superior to the previous best solutions given in Tiwari and Chakraborti (2006), thus demonstrating the better behaviour of the new approach.

4.2 *Single-objective optimisation*

The application of MOEAs that generate solutions to the 2DSP according to two different optimisation criteria has a major advantage for potential customers: such approaches provide a set of solutions offering a range of trade-offs between the two objectives, from which clients can choose according to their needs as dictated by, for example, raw material costs or even production time restraints. However, dealing with more than one optimisation objective need not imply a reduced solution quality simply because the problem in question is much more complex. On the contrary, by considering both the length and number of cuts, the approaches proposed here derive solutions with wastage levels similar to those in most previous approximations which merely seek to optimise the overall length.

Table 2 shows the best required overall length obtained for certain 2DSP test instances available in the literature [Wang and Valenzuela (2001)]. The table shows the length achieved by three different heuristics and a genetic algorithm proposed by Mumford-Valenzuela *et al.* (2004), all of which deal with the single-objective formulation of the 2DSP. The last column of the table represents the best solution - overall length - achieved by the approach proposed here when the second objective - the number of cuts - is not considered. For smaller test instances, the multi-objective approach provides better solutions than some heuristics that have been

Table 2. Comparison of multiple and single objective approaches.

Problem	NFDH	FFDH	SPLIT	GA	NSGA-II
Nice.25	133	118	138	106	106
Nice.50	120	119	134	105	109
Nice.100	112	111	137	108	111
Nice.200	110	108	138	108	113
Nice.500	108	107	139	108	126
Path.25	132	120	136	106	101
Path.50	143	131	154	106	103
Path.100	120	109	137	109	108
Path.200	128	116	138	112	121
Path.500	112	105	141	136	147

[a]The best values obtained are shown for each approach considered.

[b]All the results are rounded to the nearest integer.

specifically designed to deal with the single-objective formulation of the problem. Length solutions given by NSGA-II in the smaller test problems are similar to those obtained by a simpler single-objective genetic algorithm. But when the size of the instances increases, the specific single-objective approximations tend to behave better, thanks to their simplicity and their single optimisation considerations.

5. Conclusions

This work has shown the validity of MOEAs in dealing with the multi-objective formulation of the two-dimensional strip packing problem. Such a formulation is highly suited to the real-world problem appearing in some production and manufacturing industries. The consideration of both the number of cuts involved and the amount of raw material used could imply significant cost savings for companies. When the raw material is sufficiently cheap, a solution maximising the number of cuts, and thus the productivity of the machinery, is usually more convenient. If the raw material is very expensive, a solution minimising the length of the stock sheet required is actually the best option. This way only one is execution is needed, and afterwards the decision maker has to select the most appropriate solution, depending on the specific demands involved at the moment. When applying single-objective approaches, such specific demands must be known beforehand in order to combine the different objectives in a suitable manner. If the demands vary, new executions will naturally be needed. This availability of solution diversity is the most important benefit of the approaches proposed here.

Instead of developing some of the existing best-known MOEAs from scratch, a customisable multi-objective framework based on evolutionary algorithms [León *et al.* (2007, 2008)] was used. The results obtained with three different MOEAs demonstrate the validity of evolutionary strategies in this kind of multi-objective real-world problems. Such results clearly improve on those obtained previously in Tiwari and Chakraborti (2006), which is the only known and first attempt at the multi-objective guillotinable formulation of the problem. Since the gene representation used allows for both guillotine and non-guillotine cutting processes, the number of cuts required may vary depending on the features of the machines available. The computational study shows that whenever non-guillotine cuts are possible, i.e. the machinery allows for this kind of more sophisticated cutting process, the number of cuts required to accomplish the job can be significantly reduced. Moreover, by considering both the length and number of cuts, the approaches proposed yield solutions with wastage levels similar to those in most previous approximations which merely seek to optimise the overall length.

However, those techniques which are able to implicitly handle multiple objectives are much more complex and require greater computational effort. For some large problem instances, MOEAs-based approaches are either not able to obtain the optimal solution set, or they need excessive computational resources to do so. In future work, further analysis must be performed on the behaviour of other existing MOEAs and also on the design and test of other mutation and crossover operators that may lead to more efficient schemes. As an alternative, the incorporation of some interactive decision making during the MOEAs execution may allow for some non-interesting areas of the solution space to be discarded, while still enabling a diversity of solutions to be obtained in the most suitable regions. In addition, it has been demonstrated that using non-guillotine cuts highly reduces the number of cuts required. If the industrial machines available allow for this kind of cut, it would be recommended to focus solely on this specific problem without having to consider only guillotinable-type constructions. This would make it possible to define a completely different gene representation and genetic operators. Although the problem would become much more restrictive, a new range of implementations is available for exploration.

6. Acknowledgements

This work was funded by the EC (FEDER) and the Spanish Ministry of Education and Science as part of the 'Plan Nacional de I+D+i' (TIN2005-08818-C04-04). The work of Gara Miranda was developed under grant FPU-AP2004-2290 while the work of Jesica de Armas was funded by grant FPU-AP2007-02414.

References

Bekrar, A., Kacem, I. and Chu, C., 2007. A Comparative Study of Exact Algorithms for the Two Dimensional Strip. *Journal of Industrial and Systems Engineering*, 1 (2), 151–170.

Bortfeldt, A., 2006. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research*, 127 (3), 814–837.

Coello, C.A., 1999. An Updated Survey of Evolutionary Multiobjective Optimization Techniques: State of the Art and Future Trends. *Congress on Evolutionary Computation.*, Vol. 1 IEEE Press, 3–13.

Coello, C.A., Lamont, G.B. and Veldhuizen, D.A.V., 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation.

Cowton, C.J. and Wirth, A., 1993. On the economics of cutting tools. *International Journal of Production Research*, 31 (10), 2441–2446.

Dagli, C.H. and Tatoglu, M.Y., 1987. An approach to two-dimensional cutting stock problems. *International Journal of Production Research*, 25 (2), 175–190.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6, 182–197.

Dimopoulos, C., 2006. Multi-objective optimization of manufacturing cell design. *International Journal of Production Research*, 44 (22), 4855–4875.

Dowsland, K.A. and Dowsland, W.B., 1992. Packing Problems. *European Journal of Operational Research*, 56 (1), 2–14.

Dyckhoff, H., 1990. A Typology of Cutting and Packing Problems. *European Journal of Operational Research*, 44 (2), 145–159.

Eiben, A.E., 1998. *Handbook of Evolutionary Computation*. IOP Publishing Ltd. and Oxford University Press.

Fonseca, C.M. and Fleming, P.J., 1995. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3, 1–16.

Gagné, C. and Parizeau, M., 2006. Genericity in Evolutionary Computation Software Tools: Principles and Case Study. *International Journal on Artificial Intelligence Tools*, 15 (2), 173–194.

Garrido, P. and Riff, M.C., 2007. Collaboration Between Hyperheuristics to Solve Strip-Packing Problems. *In*: S.B.. Heidelberg, ed. *Foundations of Fuzzy Logic and Soft Computing.*, Vol. 4529 of *LNCS*, 698–707.

Goldberg, D.E. and Lingle, J.R., 1985. AllelesLoci and the Traveling Salesman Problem. *Proceedings of the 1st International Conference on Genetic Algorithms*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc., 154–159.

Hopper, E. and Turton, B.C.H., 2001. A Review of the Application of Meta-Heuristic Algorithms to 2D Strip Packing Problems. *Artificial Intelligence Review*, 16 (4), 257–300.

Horn, J. and et al., 1994. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation.*, Vol. 1, 82–87.

Illich, S., While, L. and Barone, L., 2007. Multi-objective strip packing using an evolutionary algorithm. *IEEE Congress on Evolutionary Computation*. IEEE Computer Society, 4207–4214.

Knowles, J., Thiele, L. and Zitzler, E., A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. , 2006. , TIK Report 214, Computer Engineering and Networks Laboratory (TIK), Zurich, Switzerland.

León, C., Miranda, G. and Segura, C., 2007. Parallel Skeleton for Multi-Objective Optimization. *Genetic and Evolutionary Computation Conference*. ACM, p. 906.

León, C., Miranda, G. and Segura, C., 2008. Parallel Hyperheuristic: A Self-Adaptive Island-Based Model for Multi-Objective Optimization. *Genetic and Evolutionary Computation Conference*. ACM, 757–758.

Liefooghe, A., Basseur, M., Jourdan, L. and Talbi, E.G., 2007. ParadisEO-MOEO: A Framework for Evolutionary Multi-objective Optimization. *Evolutionary Multi-Criterion Optimization.*, Vol. 4403 of *LNCS* Springer, 386–400.

Lodi, A., Martello, S. and Monaci, M., 2002. Two-Dimensional Packing Problems: A Survey. *European Journal of Operational Research*, 141 (2), 241–252.

Mumford-Valenzuela, C.L., Vick, J. and Wang, P.Y., 2004. Heuristics for large strip packing problems with guillotine patterns: an empirical study. *In*: *Metaheuristics: computer decision-making.*, 501–522 Norwell, MA, USA: Kluwer Academic Publishers.

Nearchou, A.C., 2007. Multi-objective balancing of assembly lines by population heuristics. *International Journal of Production Research*, 46 (8), 2275–2297.

Ono, T. and Ikeda, T., 1998. Optimization of two-dimensional guillotine cutting by genetic algorithms. *In*: H.J. Zimmermann, ed. *European Congress on Intelligent Techniques and Soft Computing.*, Vol. 1, 7–10.

Padmanaban, K.P. and Prabhaharan, G., 2008. Dynamic analysis on optimal placement of fixturing elements using evolutionary techniques. *International Journal of Production Research*, 46 (15), 4177–4214.

Schaffer, J.D., 1985. Multiple objective optimization with vector evaluated genetic algorithms. *In*: J.J. Grefenstette, ed. *International Conference on Genetic Algorithms and Their Applications.*, 93–100.

Song, X., Chu, C., Nie, Y. and Bennell, J., 2006. An iterative sequential heuristic procedure to a real-life 1.5-dimensional cutting stock problem. *European Journal*

of Operational Research, 175 (3), 1870–1889.

Srinivas, N. and Deb, K., 1994. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2 (3), 221–248.

Sweeney, P.E. and Paternoster, E.R., 1992. Cutting and Packing Problems: A Categorized, Application-Orientated Research Bibliography. *Journal of the Operational Research Society*, 43 (7), 691–706.

Tiwari, S. and Chakraborti, N., 2006. Multi-objective optimization of a two-dimensional cutting problem using genetic algorithms. *Journal of Materials Processing Technology*, 173, 384–393.

Wang, P.Y. and Valenzuela, C.L., 2001. Data set generation for rectangular placement problems. *European Journal of Operational Research*, 134 (2), 378–391.

Ye, M. and Zhou, G., 2006. A local genetic approach to multi-objective, facility layout problems with fixed aisles. *International Journal of Production Research*, 45 (22), 5243–5264.

Zitzler, E., 1999. Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Thesis (PhD). Swiss Federal Institute of Technology (ETH), Zurich Supervisors: L. Thiele and K. Deb.

Zitzler, E., Deb, K. and Thiele, L., 2000. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8 (2), 173–195.

Zitzler, E. and Künzli, S., 2004. Indicator-Based Selection in Multiobjective Search. *VIII Conference on Parallel Problem Solving from Nature.*, Vol. 3242 of *LNCS* Springer, 832–842.

Zitzler, E., Laumanns, M. and Thiele, L., 2002. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. *Evolutionary Methods for Design, Optimization and Control*. Barcelona, Spain: CIMNE, 19–26.

Zitzler, E. and Thiele, L., An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. , 1998a. , Technical report 43, Computer Engineering and Networks Laboratory (TIK), Zurich, Switzerland.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M. and Grunert da Fonseca, V., 2003. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7 (2), 117–132.

Zitzler, E. and Thiele, L., 1998b. Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. *Parallel Problem Solving from Nature.*, Vol. 1498 of *LNCS* Springer, 292–301.