A GENETIC SYMBIOTIC ALGORITHM APPLIED TO THE CUTTING STOCK PROBLEM WITH MULTIPLE OBJECTIVES

Rodrigo Rabello Golfeto
rodrigo.golfeto@gmail.com
Fluminense Federal University - UFF

Antônio Carlos Moretti
moretti@ime.unicamp.br
University of Campinas - UNICAMP

Luiz Leduíno de Salles Neto
luiz.leduino@unifesp.br
Federal University of São Paulo - UNIFESP

Abstract

This work presents a genetic symbiotic algorithm to solve the one-dimensional cutting stock problem with multiple objectives. We considered two important objectives for an industry (1) cost of trim loss and (2) cost of setup. We use a symbiotic relationship, between the population of solutions and the population of cutting patterns, together with a niche strategy to obtain an approximation of the Pareto-front. The evolutionary approach promotes a significant diversity in the population, which is a desirable condition to solve a multi-objective combinatorial optimization problem. Results of the computational experiments with instances from a chemical-fiber company and with random instances are reported.

Keywords: Cutting Stock Problem, Genetic Symbiotic Algorithm, Multi-objective Combinatorial Optimization.

1 Introduction

The cutting stock problem (CSP) is a classical problem in the area of Operations Research. It can be stated as the problem of finding the best way of cutting demanded items from a stock rolls of length W such that the trim loss is minimized and the total demand is satisfied. This problem arises in the production of paper (Haeesler (1976), Diegel (1988)), steel (Eilon (1960), Wascher *et al.* (1985)), window manufacturing (Stadler (1990)), etc. A CSP can be decomposed into two distinct subproblems: (1) the generation of the cutting patterns and (2) the determination of its frequency in the final solution.

Cutting Stock Problems is one of the first applications of Operations Research, it was first studied by Kantorovich in the thirties. Problems of the same nature were treated by Paull and Walter (1954), Metzger (1958) and Eilon (1960). However, as observed by Dowsland and Dowsland (1992), the methods used at that time were only appropriate for small problems. Gilmore and Gomory (1961, 1963) developed a clever procedure to deal with large problems. Haessler (1975) was the first work to treat the nonlinear cutting stock problem, considering the reduction of trim loss and the setup number of the cutting machine.

A reasonable goal to be met in a industry is to minimize the number of master rolls (i.e., objects) used to produce the demanded items. In some cases, minimize the number of objects used are not

AMO - Advanced Modeling and Optimization. ISSN: 1841-4311

the only goal for the manager. In fact, when we have a large demand that needs to be met in a short period of time, the number of machine setup takes a relevant importance. Wascher (1990) criticizes the traditional planning models for cutting stock problems, he says that they do not consider all the factors of the process which tend to have effects on the company's profit, such as the setup number, or the amount of stocked material caused by the various cutting process, etc. However, Wascher does not include the setup in his interactive multi-objective approach.

Combinatorial problems involving setup costs are known to be very hard to solve. In particular this problem presents two conflicting objectives: (1) Minimize the trim loss and (2) the total number of setup used. The difficulty on solving this problem is due to the fact that the objective function besides being nonlinear it is also discontinuous. Recent approaches can be found in Diegel *et al.* (1996), Foester *et al.* (2000), Vanderbeck *et al.* (2000), Belov *et al.* (2003), Umetani *et al.* (2003,2006), Yanasse *et al.* (2006), Lee (2007) and Moretti *et al.* (2008). We have not seen a multi-objective approach considering, trim loss and setup, in a cutting stock problem with any number of demanded items. In this work we present a Genetic Symbiotic Algorithm (Symbio) to multi-objective cutting stock problem (MoCSP), where the objectives functions are trim loss and setup.

The outline of this paper is as follows: Section 2 introduces the cutting stock problem. Section 3 gives a general introduction of the multi-objective optimization and the Multi-objective Cutting Stock Problem (MoCSP). Section 4 presents the description of our approach: a Genetic Symbiotic Algorithm (Symbio) to MoCSP. Experimental results are presented and analyzed in Section 5. Finally, in Section 6, we conclude the paper and give an overview of possible future work.

2 One-Dimensional Cutting Stock Problem

The Standard Cutting Stock Problem is characterized by cutting stock rolls of size W (called *objects*) into smaller pieces of size w_i (where $W > w_i, i = 1, 2, \ldots m$), with the objective of satisfying the demand d_i for each one of these m items. According to the typology of Wascher *et al.* (2007) this problem is classified as SSSCSP. Each combination of items in an object is called *cutting pattern* and each changing of a cutting pattern has a *setup cost* to prepare the cutting machine. The setup number has a great importance when we need to meet a large demand of items in a short period of time. We want to find a trade-off between the setup number and the trim loss. The mathematical model to minimize cost of the trim loss and cost of the setup number can be stated as:

$$\text{Minimize} \quad c_1 \left(W \times \sum_{j=1}^{n} x_j - \sum_{i=1}^{m} w_i d_i \right) + c_2 \sum_{j=1}^{n} \delta(x_j)$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq d_i, && i = 1, ..., m. \\ & x_j \in \mathbb{N}, && j = 1, ..., n. \end{aligned}$$

where c_1 is the cost of the trim loss; a_{ij} is the number of times item i appears in the j^{th} cutting pattern; x_j is the number of objects processed with the cutting pattern j; c_2 is the setup cost and $\delta(x_j) = \begin{cases} 1 & \text{if } x_j > 0, \\ 0 & \text{if } x_j = 0. \end{cases}$

In the literature, Diegel *et al.* (1996) were the only to mention real-life values for c_1 and c_2. According to Diegel, an exact relation between c_1 and c_2 depends on several factors as: demand, deadlines, labor costs, etc. The major advantage of a multi-objective approach is to give a set of solutions which are not evaluated by a common scalar function, i.e., the optimization process is, in theory, not biased toward a particular type of solution that depends on c_1 and c_2.

3 The Multi-objective Cutting Stock Problem

A Multiple Objective Program (MOP) can be written as:

$$\begin{array}{ll} \text{Min} & \{f_1(x) = z_1\} \\ \text{Min} & \{f_2(x) = z_2\} \\ \vdots & \vdots \\ \text{Min} & \{f_k(x) = z_k\} \\ \\ \text{s. t.} & x \in S \end{array}$$

where: k is the number of objectives; $f : R^n \rightarrow R^m$ is the vector objective function; $S \subset R^n$ is the feasible region; z_i is the criterion value; z is the criterion vector.

Definition [Dominance] Let z^1, $z^2 \in \mathbb{R}^k$ be two criterion vectors. Then, z^1 dominates z^2 if and only if $z^1 \leq z^2$ and $z^1 \neq z^2$ (that is, $z_i^1 \leq z_i^2$ for all i and $z_i^1 < z_i^2$ for at least one i).

A criterion vector is non-dominated if it is not dominated by any other feasible criterion vector. Let $Z = \{z \in \mathbb{R}^k : z = f(x), x \in S\}$ be the feasible region in criterion space. We use the notation $z^1 \succ z^2$ to indicate that the vector z^1 dominates the vector z^2. Or equivalently $z^2 \prec z^1$.

Definition [Non-dominance] Let $\bar{z} \in Z$. Then, $\bar{z}$ is non-dominated if and only if does not exist another $z \in Z$ such that $z \leq \bar{z}$ and $z \neq \bar{z}$. Otherwise, $\bar{z}$ is a dominated criterion vector.

Although the idea of dominance refers to vectors in the criterion space, the idea of efficiency refers to points in decision space.

Definition [Efficiency] A point $\bar{x} \in S$ is efficient if and only if there does not exist another $x \in S$ such that $f(x) \leq f(\bar{x})$ and $f(x) \neq f(\bar{x})$. Otherwise, $\bar{x}$ is inefficient.

Therefore, a point $\bar{x}$ is efficient if its criterion vector is not dominated by any criterion vector of other point in S. That is, it is not possible to move in a feasible direction to decrease one of the objectives without necessarily increasing at least one of the other objective values. The term efficiency is also known as Pareto optimal and the curve in the space of the objective functions formed by non-dominated vectors that are in the Pareto optimal set is called the Pareto-front.

Solving a multi-objective optimization problem is to find the set of efficient solutions. In the present work we are particularly interested on finding the set of efficient solutions X^* of the MoCSP:

$$(MoCSP) \begin{cases} x^* = eff_x \ f(x) = (f_1(x), f_2(x)) \\ \text{s. t. } x \in X \end{cases}$$

where

- $f_1(x) = \left(W \times \sum_{j=1}^{n} x_j - \sum_{i=1}^{m} w_i d_i \right) \equiv$ trim loss
- $f_2(x) = \sum_{j=1}^{n} \delta(x_j) \equiv$ setup.
- $X = \{x \in Z^n, \ x \geq 0 : y_i = \sum_{j=1}^{n} a_{ij} x_j - d_i \geq 0, i = 1, ..., m\}$.

The MoCSP is a multi-objective combinatorial optimization (MOCO) which differs from the traditional single objective optimization in several ways (see Ehrgott (2004)):

- The usual meaning of the optimum makes no sense in the multiple objective case because a solution that optimizes all objectives simultaneously does not exist in general.

- The identification of a best compromise solution requires to take in account the preferences expressed by the decision maker.
- The MOCO problems can be very hard to solve exactly (see, Ehrgott (2000)). Even if they are derived from easy single objective optimization problems.

It is known that the multi-objective linear programming problem

$$\text{Min}\{(c^1x, c^2x, ..., c^qx) : Ax = b, x \geq 0\, x \in \Re^n\}$$

the set of efficient solutions is exactly the set of solutions that can be obtained by solving the linear programming problem $Min\ \{\sum_{j=1}^{q} \lambda_j c^j x : Ax = b, x \geq 0\}$, where $\sum_{j=1}^{n} \lambda_j = 1, \lambda_j \geq 0$. But the discrete structure of the MOCO problem makes this result invalid, since there exist efficient solutions, which are not optimal for any weighted sum of the objectives.

Let

$$Z^{\geq} = \text{Convex hull of } [Z^* \oplus \{z \in R^k \text{ such that } z \geq 0\}]$$

where Z^* is the Pareto-Front and $\oplus$ means the set addition and consider the following definitions.

Definition [Unsupported] Let $z \in Z^*$. Then, if z is on the boundary of $Z^{\geq}$, z is a supported non-dominated criterion vector. Otherwise, z is an unsupported (convex dominated) non-dominated criterion vector.

Inverse images of supported non-dominated criterion vectors are said to be supported efficient points (in the decision space) and inverse images of unsupported non-dominated criterion vectors are said to be unsupported efficient point (in the decision space).

Unsupported non-dominated criterion vectors are dominated by some convex combination of other non-dominated criterion vectors. In multiple objective combinatorial optimization the unsupported non-dominated criterion vectors are not rare. In this case it is not possible to generate unsupported criterion vectors using the weighted sum strategy. Next example shows this situation in a MoCSP problem.

Example 1: Consider the data

- $W = 20$;
- $m = 4$;
- $w = (w_1, w_2, w_3, w_4) = (10, 6, 5, 4)$;
- $d = (d_1, d_2, d_3, d_4) = (600, 153, 300, 15)$;

To make the notation easier, let us represent the j^{th}-cutting pattern as a vector $a_j \in Z^m$, where the i^{th} component is the quantity of items i produced by one application of the cutting pattern.

Enumerating all the solutions we can conclude that have three efficients solutions.

Efficient Solution 1:

- Setup = 4;
- Trim Loss = 102;
- Cutting Pattern 1: $a_1 = (2, 0, 0, 0)$ with $x_1 = 300$;
- Cutting Pattern 2: $a_2 = (0, 3, 0, 0)$ with $x_2 = 51$;
- Cutting Pattern 3: $a_3 = (0, 0, 4, 0)$ with $x_3 = 75$;
- Cutting Pattern 4: $a_4 = (0, 0, 0, 5)$ with $x_4 = 3$.

Efficient Solution 2:

- Setup = 3;
- Trim Loss = 542;
- Cutting Pattern 1: $a_1 = (2, 0, 0, 0)$ with $x_1 = 300$;
- Cutting Pattern 2: $a_2 = (0, 1, 2, 1)$ with $x_2 = 150$;
- Cutting Pattern 3: $a_3 = (0, 3, 0, 0)$ with $x_3 = 1$.

Efficient Solution 3:

- Setup = 2;
- Trim Loss = 582;
- Cutting Pattern 1: $a_1 = (2, 0, 0, 0)$ with $x_1 = 300$;
- Cutting Pattern 2: $a_2 = (0, 1, 2, 1)$ with $x_2 = 153$.

Looking at Figure 1, we see that the Pareto-Front is nonconvex, therefore, the weighted sum strategy will not be able to compute all efficients solutions.

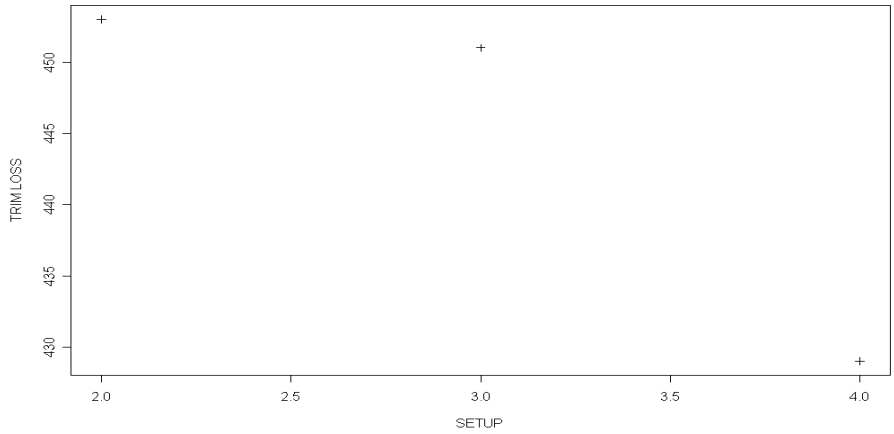

Figure 1: Example 1

In general, we can not generate all efficient points with the approach

$$\text{Min}\{\sum_{i=1}^{k} \lambda_i f_i(x) \text{ such that } x \in S\},$$

in this case the use of an evolutionary method for MoCSP can be more appropriate.

In the literature we find only one paper that works with multi-objective cutting stock problem involving setup (see, Kolen (2000)). But, this paper works only with small instances

- the number of different items varies between 2 and 8;
- the width of demanded items is less than 1/3 of the size W of the stock.

On the other hand, our method is tested in instances coming from a chemical fiber company in Japan (see Umetani *et al.* (2003)) where:

- the number of different items varies between 4 and 20;
- the width of demanded items varies between 5,5% and 40% of W.

Also, we tested the method in 10 instances generated by CUTGEN1 (see, Gau 2000), where the number of items is between 10 and 20. The size of the items is between 1 and 80% of the size of W.

Also, in Kolen (2000) the number of processed objects is given and it is constant.

In Section 5 we compared our results with the results obtained by Lee (2007) and Umetani *et al.* (2003, 2006).

4 Genetic Symbiotic Algorithm

Genetic algorithm (GA) have theirs basis in the works of Holland (1962), Bremermann (1962) and Fraser (1957). However, only in the work of Holland (1975) the genetic algorithm was, in fact, introduced. In the past decade it became a promising method for solving optimization problems.

GA is a particular class of evolutionary computation (EC) that uses techniques inspired in the theory of evolution, where the stronger organisms have more chances to reproduce and pass their characteristics to the next generation. This is used as paradigm to solve problems.

Some concepts associated with genetic algorithm are:

Fitness: indicates the level of adaptability of one individual to environment.

Genes: functional blocks of DNA.

Genome: a collection of genes of one individual.

Selection: is the mechanism responsible to select the best individuals in a population. The most used types of selection are: tournament, elitism, roulette.

Crossover: also called recombination, it is a genetic operator used to combine two individuals to generate a new individual. There are many types of crossover, the most used are: One-point, Two-point and Uniform.

Mutation: The mutation operator consists of a random modification, usually with low probability, in the value of an or more genes of the an individual. This is an operator executed after the crossover.

Coevolution: it is a technique where several populations are used to increase the diversity of the candidate solutions through a migration mechanism. This technique will be discussed in details in Subsection 4.2.4.

We worked with a biological concept called symbiosis, where it is possible to ally abilities of different individuals with the objective to solve a common problem. In our method, we applied coevolution techniques with the theory of niches to obtain an approximation of the Pareto-Front.

4.1 Symbiosis

In Allaby (1998), the term symbiosis is defined as a general term describing the situation where dissimilar organisms live together in a close association. As originally defined, the term embraces all types of mutualism and parasitic relationship. Its modern use is often restricted to mutually beneficial species interaction. Mutualism is defined as an interaction between members of two species

with benefits for both. Pianka (1994) shows some examples of interactions among species as, for example, nectar-feeding birds and flowering plants, ants and plants, birds and buffalos.

The works of Eguchi *et al.* (2003), Hirasawa *et al.* (2000) and Mao *et al.* (2000) also simulate the symbiosis relationships. However, in theirs algorithms each individual of the population is treated like one different specie that develops a symbiosis relationship with another individual. In Hirasawa *et al.* (2003), the authors give one example of one possible relationship: "if individual i exists near individual j and the fitness of individual i is greater than that of individual j, then individual i exploits individual j".

The genetic symbiotic algorithm (GSA), also called cooperative algorithm (Potter 1997, Kim *et al.* 2000), basically decomposes the problem in n subproblems using n different species. Dividing the problem into n distinct populations we can solve the problem utilizing simple structures, that working together can be more powerful than complex structures. Kim *et al.* (2001, 2006) proposed an endosymbiotic evolutionary algorithm for optimization where the idea basic is the incorporation of the evolution of the eukaryotic cells (Margullis 1981) into the existing symbiotic algorithms. In this approach when an individual meets a partner giving a high fitness, the whole combination envolves for some time without changing its partner. Tsujimura *et al.* (2001) presented um symbiotic genetic algorithm to the job shop schedulling. Chang *et al.* (2002) presented a symbiotic evolutionary for dynamic facility layout problem.

4.2 Implementation

The main difference between the GSA versus the classical genetic algorithm (CGA), when applied to CSP, is the ability of constructing cutting patterns and candidate solutions at the same time. That is, the cutting patterns are not a part of the solution, although the solution is dependent of the cutting patterns, we can work with them separately.

Khalifa *et al.* (2006) solve the CSP using a genetic algorithm, where the genes of each solution are processed in pairs and the first gene of each solution represents the frequency of the pattern which is represented by the second gene. In our implementation the second gene represents an individual of the population of patterns. For example, Figure 2 shows patterns 37, 11 and 32 with theirs frequencies 2, 4 and 5, respectively.

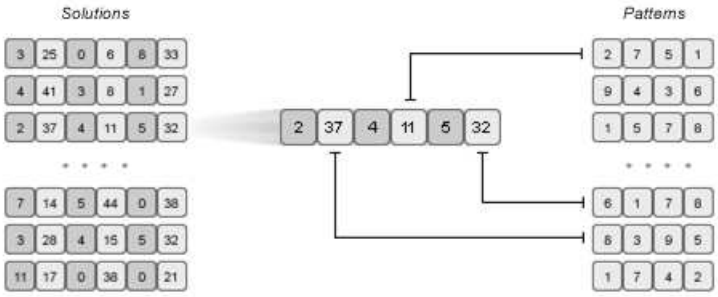

Figure 2: Structures of the genes.

The biological relationship that is closer to our implementation is the mutualism, since both individuals can take advantage of the relationship. The relationship in this case represents the level of adaptability to the environment.

In the next subsections we explain the structure of each population, the niches and the coevolution system. At this point is important to establish that we call the individuals of the first population by solutions and the individuals of the second population by patterns. Observe that we work with several populations of solutions and cutting patterns, we consider each group of them as an association. Therefore, we have two different populations (solutions and cutting patterns) in each association.

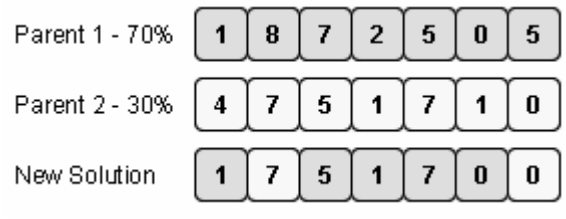

Figure 3: Uniform crossover

The algorithm ends when the maximum number of 10,000 generations is reached.

4.2.1 Individual-Solutions

Below, we describe the parameters of the first specie (i.e., solutions):

- **Population size:** $3,000$ individuals, in spite of the high computational cost, to work with a population of this size promotes a great diversity, and this is important to avoid an early convergence of the algorithm;
- **Type of selection:** elitism, top 1/3 of the best individuals;
- **Crossover rate:** 2/3, this value was determined by trial-and-error. We believed that this represents a good trade off between preservation and generation of new individuals ;
- **Crossover type:** uniform, 70% of chance for the best individual (see figure 3);
- **Mutation rate:** we calculate the probability of 2 genes mutating; that is, if the individual has k genes, the probability of each gene mutating is $2/k$ (see figure 4).

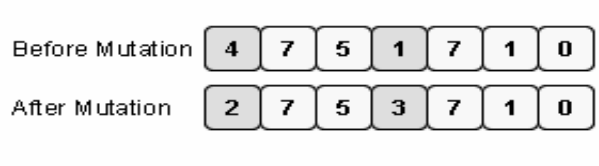

Figure 4: Mutation of an individual/solution

To determine the size of the DNA chain (i.e., the number of genes of an individual), we estimate the maximum number of setups that one problem might have. We choose this number to be equal m since the problem of minimizing the trim loss in a cutting plan can be written as a linear programming problem, where each constraint represents the demand of an item. The number of genes of an individual-solution is fixed as the double of the maximum number of setups.

The frequency of each pattern, which is represented by the odd gene, has bounds. This restricts the size of the search region. The lower bound is fix to zero, and the upper bound is fixed as the biggest demand in the cutting plan. This is done to allow just one type of item in a cutting pattern.

The fitness of each objective function is computed as follows:

- $f_1^*(x) = W \times \sum_{j=1}^{n} x_j - \sum_{i=1}^{m} w_i d_i + \rho$
- $f_2^*(x) = \sum_{j=1}^{n} \delta(x_j) + \frac{\sum_{i=1}^{m} w_i |\Gamma_i - d_i|}{\sum_{i=1}^{m} w_i d_i} + \rho$

where: Γ_i represents the amount of items of type i in the solution; ρ is a penalty if the solution is infeasible.

The term Γ_i, in function f_2^*, validate solutions with any or small excess of production, allowing two individual-solutions to be considered non-dominated with the same setup, increasing the diversity of the population. However, in the last generation the parameters Γ_i are suppressed of function f_2^* and only the non-dominated solutions are given in the final solution.

To avoid an early convergence of the method, two solutions with same values of $f_1^*(x)$ and $f_2^*(x)$ are not allowed. When this happens one of the solutions receives a great penalty. Besides, at every 100 generations we generate 100 randomly individuals and place them in the elite list, replacing the 100 worst solutions. This is another strategy to increase the diversity in the population.

The values of $f_1^*(x)$ e $f_2^*(x)$ are used to determine the strength fitness S_i (see Zitzler *et al.* (2001)) for each individual i in the population of solutions. This is done to rank the solutions using the dominance concept. The value of S_i is calculated as

$$S_i = |\{j, j \in Q, \text{ such that } i \prec j\}|$$

where Q is the set of the individuals in the population of solutions.

In fact, with this definition, all solution with S_i equal to zero will be considered non-dominated.

4.2.2 Individual-Patterns

The parameters of the population of cutting patterns are different of the parameters of the population of solutions:

- Population size: 900 individuals;
- Type of selection: elitism, 2/3 of the best individuals;
- Crossover rate: 1/3, obtained in an experimental way;
- Crossover type: 2 points (see figure 5);
- Mutation rate: 1% of probability of at the most one gene mutates (see figure 6).

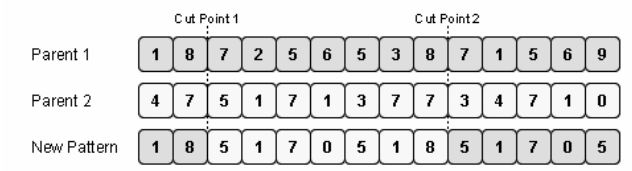

Figure 5: Two-point crossover

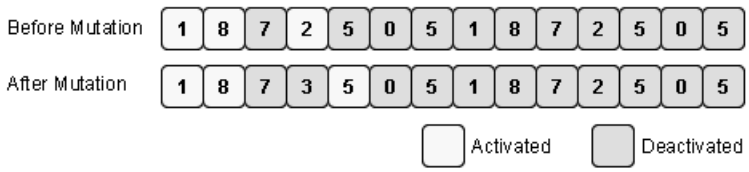

Figure 6: Mutation of an individual/pattern

The length of the DNA chain is equal to the $\lfloor W/\min_{1\leq i\leq m}\{w_i\}\rfloor$ to guarantee that it is possible to create a pattern using only one item type.

However, some of cutting patterns do not use all genes available in the DNA chain, since we consider the items in the cutting pattern until the sum of their lengths do not exceed the size of the object.

We calculate the fitness of an individual-pattern in the elite list according to the following procedure

```
For i = 1 to EliteSolutions
  For each pattern j into the solution i Do
    If S(i) = 0 Then
      FitnessPattern(j) = FitnessPattern(j) + 10
    Else
      FitnessPattern(j) = FitnessPattern(j) + 1 + (1 / i)
    End if
  End For
End For
```

Besides, we keep the individual-patterns associated with the best 20 individual-solutions inside each niche.

4.2.3 Niches

Since the number of comparisons necessary to calculate S_i grows too fast in relation the number of individuals in the population of solutions, a niche strategy is adopted to make the algorithm more efficient.

To reduce this cost, each value of f_2 (representing the setup) is used as one distinct niche. In other words, we put together the individuals with the same setup number and we use the concept of local dominance to calculate S_i.

However, this strategy may cause a problem. Given two solutions, α_1 and α_2, where $\alpha_1 \succ \alpha_2$. If $f_2(\alpha_1) \neq f_2(\alpha_2)$ then the two solutions are not compared and α_2 maybe considered as a non-dominated solution. To avoid this, we compare all solutions in niche p (where $p \leq m$ is equal the setup number) with the best solutions of niches $p-1, p-2$.... In this case if the solution of niche n is dominated by the best solution of an inferior niche, the corresponding S_i is penalized by 10 points.

After determining S_i, we put the individual-solutions in a fitness ascending order and include them again in the niches. The position of the individual-solution in the niche is important, since we use the value of the fitness to calculate the roulette of the 5 top individual-solutions in each niche. The best individual-solution in each niche has the value of fitness equal 2, while the other individual-solutions have their fitness lower than 2.

The strategy of developing solutions in all niches is important, since if no change happens in the fitness of the dominated solutions in the niches of high setups, the solutions in these niches are probably lost, and we do not want to lose information, since, it can be hard to recover it.

The niche is useful to establish which solutions are recombined. It is easier to see that the crossover in the inner niche has a greater chance to generate improved individual-solutions. Therefore, we establish that 70% of crossover operations are made with individuals of the same niche. Another important function of niche is to maintain the diversity in population, since we are preserving individuals with different setups then the random crossovers are used to maintain the diversity in the population.

Although we perform 70% of crossovers in the inner niches, we still have to establish a criterion to choose which niches generate new individuals. To do so, we use the fraction between the number of dominated solutions and the number of solutions generated by the niche and, after that, we use a roulette to choose the niche.

The experiments suggest that eliminating individual-solutions by the distance between them, an common technique in multi-objective algorithms, is not efficient in this problem and, furthermore,

using another notion of distance, like the difference between the demands or the cutting patterns, is too expensive (in terms of computational time). Therefore, the strategy of generating individuals from the niches less promising is valid to maintain the diversity in the population.

4.2.4 Coevolution

We work with three different associations, each association is composed by the two populations, that work in a cooperative and independently way. The symbiotic relationship occurs between the population of solutions and the population of patterns in each association.

The migration rate among the associations is relatively small, each association receives 6 individuals each $1,000$ generations; the quantity of individuals that is transfered from one association to another is not fixed. In a cycle an association can receive 6 solutions from one association and in the next cycle 3 solutions from each association. Although, apparently small, this value is enough to produce diversity.

The main difference among the associations is the way the value of fitness function ϕ_i is computed. Another difference is that we allow infeasible solutions. Before explain this differences, we define the lower bound in the number of setups

$$l_s = \left\lceil \frac{\sum_{i=1}^{m} w_i}{W} \right\rceil$$

We can state that solutions with the number of setup less than l_s, do not exist. If a solution with this number of setup is generated it is destroyed, although we can not state that there exist solutions with l_s setups. Below, we define each association.

Association 1: This association is related to the minimization of the trim loss. Infeasible solutions will be added to niches up to its limit of 100 individuals and the function ϕ_i is given by

$$\phi_i = \frac{1}{f_1^*(x_i)} + \frac{1}{1 + S_i}$$

Therefore, the individuals with low trim loss have greater probability to generate new individuals.

Association 2: This association is related to the minimization of the number of setup. The function ϕ_i can be stated as follow

$$\phi_i = \frac{1}{f_2^*(x_i)} + \frac{1}{1 + S_i}$$

In this association the individuals with low setup is more probable to generate new individuals.

Association 3: This association is different from the previous associations for three reasons,

1. the solutions are added to the niches if there exists at least a feasible solution in the niche, that is, the niche is opened with a feasible solution;
2. the number of solutions in a niche is not limited;
3. the fitness function $\phi_i = \frac{1}{1+S_i}$ do not prioritize any objective.

4.2.5 Pseudo-Code

The steps below describe how to obtain a set of candidates to efficient points for the MoCSP:

Algorithm - Symbio

Step 1: Initialize the population of solutions with random values

Step 2: Initialize the population of patterns with random values

Step 3: Compute the f_1^* and f_2^* values of the individual-solutions

Step 4: Insert each individual-solution in its respective niche

Step 5: Compute the value of S_i for each individual-solution

Step 6: Compute the value of ϕ_i for each individual-solution based on the rules of each association

Step 7: Order the individual-solutions based on the ϕ_i values

Step 8: Insert each individual-solution in its niches based on the rules of each association

Step 9: Calculate the fitness of individual-patterns

Step 10: Select the patterns with the best fitness and put these patterns in the elite list of the first 20 individual-solutions of each niche

Step 11: Use the crossover and mutation operators to generate new individuals in each population

Step 12: If the current generation is multiple of $1,000$ migrate the individual-solutions among the associations

Step 13: If any stopping criterion is satisfied then Stop, else return to Step 3.

4.2.6 An example

In this section, we propose a small example to show how the method SYMBIO works in a cutting stock problem.

- $W = 10$;
- $m = 4$;
- $w = (w_1, w_2, w_3, w_4) = (1, 2, 3, 4)$;
- $d = (d_1, d_2, d_3, d_4) = (200, 150, 100, 100)$.

For this small example we defined the following parameters.

- **Individual-solutions**:

  Population size: 4 individuals;

  Type of selection: elitism, top 3/4 of the best individuals;

  Crossover rate: 1/4.

  Crossover type: uniform, 70% of chance for the best individual.

- **Individual-Patterns**

  Population size: 5 individuals;

  Type of selection: elitism, 4/5 of the best individuals;

  Crossover rate: 1/5;

  Crossover type: 2 points;

  Mutation rate: 1% of probability of at the most one gene mutates.

For this example, each individual-pattern is represented by 6 genes and the associations use the same population of patterns in the beginning of the process.
Patterns population: Pattern 1: (**1,2,1,4**,1,2); Pattern 2: (**1,2,3,4**,4,2); Pattern 3: (**3,3,4**,1,4,2); Pattern 4: (**1,1,1,2,3,2)**; Pattern 5: (**1,4,4**,1,2,2). Although each pattern has 6 genes, we do not use all of them necessarily. We just consider the item in the pattern if the sum of its size with the sizes of all the previous items (in the same pattern) do not exceed the size of the stock (i.e.,W). We call these items as active, the other items are not active in the pattern.

Below, we show the population of solutions for each type of association. Each individual is represented by a triple (ijk), where the i is the number of the association that the individual is related, j is the niche number of the individual and k is an arbitrary number that identifies the individual inside the niche. Since $m = 4$, each individual of the population of solutions has 8 genes, where the even gene is the frequency of the pattern stored in the next gene (odd gene).

Association 1:

- Individual 131: $(75, 1, 50, 2, 50, 3, 0, 4)$
- Individual 132: $(50, 5, 50, 1, 100, 2, 0, 3)$
- Individual 141: $(25, 1, 25, 3, 50, 4, 10, 5)$
- Individual 142: $(20, 1, 50, 2, 30, 3, 45, 4)$

Association 2:

- Individual 211: $(75, 1, 50, 2, 50, 3, 0, 4)$
- Individual 221: $(50, 5, 50, 1, 100, 2, 0, 3)$
- Individual 222: $(100, 4, 50, 5, 0, 1, 0, 5)$
- Individual 232: $(25, 4, 20, 5, 100, 2, 0, 1)$

Association 3:

- Individual 321: $(100, 1, 100, 2, 0, 3, 0, 4)$
- Individual 322: $(100, 1, 50, 2, 0, 5, 0, 3)$
- Individual 331: $(20, 3, 75, 4, 40, 5, 0, 1)$
- Individual 341: $(30, 2, 30, 3, 60, 4, 20, 5)$

Solution	f_1^*	Γ	f_2^*	S	ϕ
131	550	0.3333	3.3333	0	1.0018
132	800	0.6666	3.6666	1	0.50125
141	9910	0.0416	10004.0467	1	0.5001
142	250	0.2083	4.2083	0	1.004
211	800	0.6667	1.6667	0	1.6
221	550	0.9583	2.9583	1	0.8380
222	300	0.25	2.25	0	1.4444
231	350	0.25	3.25	1.5	0.7077
232	250	0.2083	3.2083	0	1.3116
321	800	0.6667	2.6667	1	0.5
322	300	0.25	2.25	0	1
331	150	0.125	3.125	0	1
341	200	0.16667	4.1667	0	1

Table 1: Fitness ϕ of each solution

Table 1 shows, for each individual (ljk) in the population of solutions, the values of f_1^*, $\Gamma = (\sum_{i=1}^{m} w_i \, |\Gamma_i - d_i|)/(\sum_{i=1}^{m} w_i d_i)$, f_2^*, S and ϕ, where Γ_i is the quantity of the item i produced by that solution and ϕ is the fitness of the individual computed as suggested in section 4.2.4.

Once the fitness of each individual-solution is computed, SYMBIO computes the fitness of the individual-pattern for each association as described in section 4.2.2. Table 2 shows the fitness of each pattern in each association where the column " Pattern (ij)" means pattern j in association i.

Pattern (ij)	Fitness
11	22.5833
12	21.3333
13	21.25
14	11.25
15	2.5833
21	0.2
22	20
23	0.45
24	20.25
25	20.2
31	10.25
32	10.25
33	20
34	30
35	30

Table 2: Fitness of each pattern

After computing the fitness, symbio begins the operations of crossover and mutation in each population inside the associations. Let us analyze the first population of solutions in each one of the associations. Since we use elitism with rate equals to $\frac{3}{4}$ and we have a population of size 4, only one individual will be generated. Table 3 shows, for each association, which index (ijk) will be replaced (the one with smallest fitness), which parents were chosen and what is the new individual generated by the crossover and mutation operations.

Solution replaced	Parent 1	Parent 2	New individual	
			After Crossover	After Mutation
141	131	142	141:(20, 1, 50, 2, 30, 3, 45, 4)	141:(25, 1, 50, 2, 20, 3, 45, 4)
231	222	232	231:(25, 4, 100, 5, 0, 2, 0, 1)	231:(25, 4, 100, 5, 10, 2, 0, 1)
321	331	341	321:(30, 3, 30, 4, 60, 5, 0, 1)	321:(30, 3, 20, 4, 50, 5, 0, 1)

Table 3: Crossover

Pattern replaced	Parent 1	Parent 2	New individual	
			After Crossover	After Mutation
15	11:(**1,2,1,4**,1,2)	12:(**1,2,3,4**,4,2)	15:(**1,2,3,4**,1,2)	15:(**1,2,3,3,1**,2)
21	24:(**1,1,1,2,3,2**)	25:(**1,4,4**,1,2,2)	21:(**1,1,4,1,3**,2)	21:(**1,1,4,1,3**,1)
31	34:(**1,1,1,2,3,2**)	35:(**1,4,4**,1,2,2)	31:(**1,1,4,1,3**,2)	31:(**2,1,4,1**,3,2)

Table 4: Mutation

Now, let us show the evolution of the population of patterns. We adopted the notation jk to represents the pattern k in association j. Table 4 shows the the pattern that were replaced (the one with the smallest fitness) in each association, the parents who will generate a new individual and the individual after mutation.

After the iteration,if the stopping criterion is not satisfied then the process continues. After some predefined number of iterations symbio does some migrations of individual-solution and individual-pattern from one association to another.

5 Computational Experiences

We worked with the same test problems used in Lee (2007) and Umetani (2003) plus ten more problems generated by CUTGEN1 (Gau 1995). The 40 instances used in Lee (2007) are available from Umetani (2007), they come from a chemical fiber company in Japan. The first 20 instances use stock rolls of size $W = 5180$ and the remaining 20 instances use $W = 9080$. The number of items varies from 4 to 20. As in Lee (2007), we add together the demands d_i of the items with the same width w_i. As said before, the width of the items vary from 5,5% to 40% of the width W of the stock roll. The method *ILS* (Iterated Local Search) developed by Umetani *et al.* (2003) is based on a local search heuristic that uses pricing information.

The Tables 6, 7, 8 and 9 present the comparison of results for the instances where $W = 5180$. Our method is denoted by *SYMBIO*, Lee's method is denoted by *CRAWLA* and Umetani's method is called *ILS*. Looking at the tables, we observe that Symbio presents better results in relation to the other methods when the number of demanded items is smaller and worst results, when compared with Lee's approach, when the number of items is larger. In the Tables 10, 11, 12 e 13 the results are related to the problems where $W = 9080$. We present in the respective column of each method the percentage of trim losses obtained with the respective number of different cutting patterns (setup), in the same way that was done in Lee (2007).

We analyze in detail three different instances, in all of them the size of the stock rolls is $W = 5180$. We use the dominance concept to do the graphs and we just present the best solutions of the method ILS. The problems are described below.

[Fiber06:] In this problem the SYMBIO obtained better results than the others two methods, presenting solutions that dominate all the other ones. When the setup is equal to 5, Symbio obtains the same results of ILS (Figure 7).

[Fiber13a:] After analyzing this problem, we see that the solutions of the SYMBIO and of the ILS are complementary, while the solutions of CRAWLA are totally dominated (Figure 8).

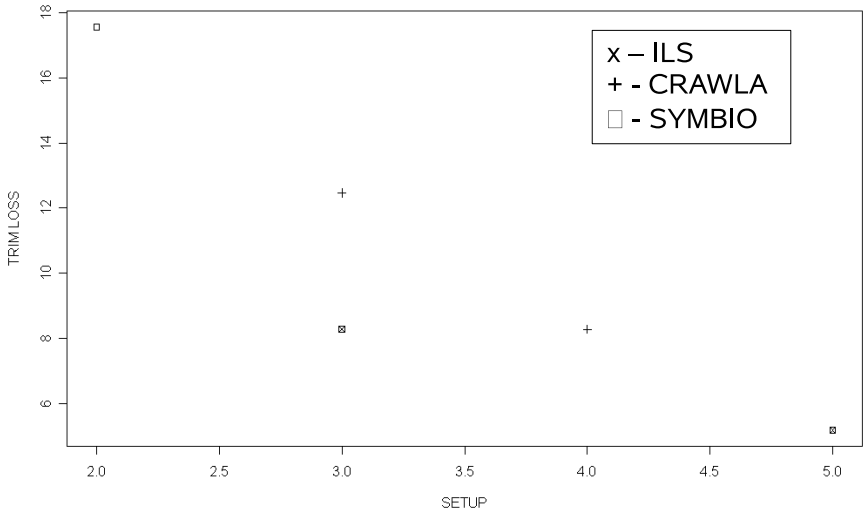

Figure 7: Fiber06.

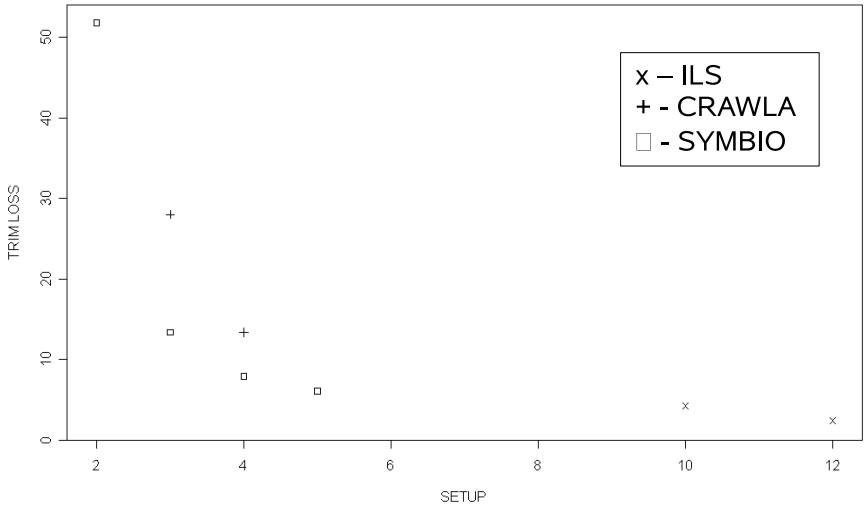

Figure 8: Fiber13a.

[Fiber26:] In this problem, despite of SYMBIO having found good solutions, all of them are dominated by the other methods (Figure 9).

We also tested symbio in 10 problems generated by CUTGEN1 (Gau 1995) to make possible the comparison done with the method presented in (Umetani 2006): a local search algorithm that uses two types of local search: (1) the 1-add neighborhood and (2) the shift neighborhood. Linear programming techniques were aggregated to the local search procedures to reduce the number of solutions in each neighborhood and to improve its performance. A sensitivity analysis technique was introduced to solve the large number of associated LP problems quickly. Umetani *et al.* (2006) compared ILS with KOMBI234, SHP and with an exact branch-and-price method (BP) proposed by Belov and

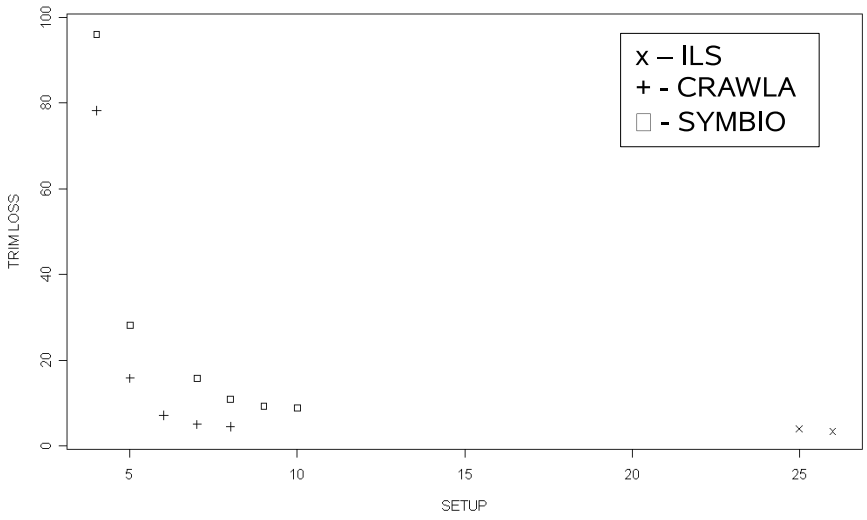

Figure 9: Fiber26.

Scheithauer (2000), which proposed a method similar to the work of Vanderbeck (2000), but, with a small number of variables. In Vanderbeck (2000), he investigates the problem of minimizing a number of different cutting patterns as a nonlinear integer programming, where a number of objects is fixed and determined after solving the problem by Gilmore-Gomory strategy. In this paper, Vanderbeck uses a Dantzig-Wolfe decomposition, developed in Vanderbeck (1999), to solve the resulting integer programming problem. Umetani *et al.* (2006) claims that the algorithm ILS obtains better quality solutions than those obtained by the SHP, KOMBI234, BP approaches.

Table 5 shows the data of these problems. Consider $W = 1000$ and $v_1 \times 1000 \leq w_i \leq v_2 \times 1000$ for $i = 1, ..., m$ and recall that m is the number of demanded items, w_i is the size of item i and d is the average demand of the items (see Gau *et al.* (1995) for details).

Instance	v_1	v_2	m	d
cutgen01	0.01	0.2	10	10
cutgen02	0.01	0.2	10	100
cutgen03	0.01	0.2	20	10
cutgen04	0.01	0.2	20	100
cutgen08	0.01	0.8	10	100
cutgen09	0.01	0.8	20	10
cutgen10	0.01	0.8	20	100
cutgen13	0.2	0.8	10	10
cutgen15	0.2	0.8	20	10

Table 5: Random Generated Instances and their parameters

Since ILS (Umetani *et al.* 2006) minimizes the setup and the number of used objects, Table 14 shows the nondominated solutions obtained by ILS and by Symbio. Both methods obtained the same solution in 3 out of the 10 problems: cutgen08, cutgen12 and cutgen15. The solutions obtained by Symbio dominated the solutions obtained by ILS in 2 problems: cutgen01 and cutgen02. In problems cutgen03, cutgen04, cutgen07, cutgen09 and cutgen10, ILS dominated Symbio.

It took 60-80 minutes to SYMBIO to run each problem and it was faster in the problems with several items. This happens because the more niches the algorithm has less comparisons it will have to do. In the small problems the algorithm possessed at the most 4 niches, while in larger problems this number is more than 10. The curve of the computational time for SYMBIO can be seen in the Figure 10.

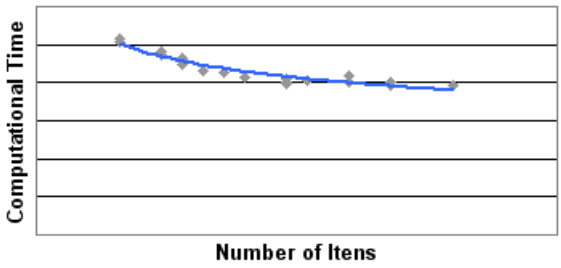

Figure 10: Computational Time.

The implementation of the method was made in FORTRAN 90/95 utilizing the Microsoft FORTRAN Power Station compiled in a micro-computer AMD SEMPRON 2300+ 1,533 MHz with 640MB of RAM. The source code is available from the web-page `http://www.otimizacao.net`.

6 Conclusions and Perspectives

The main contribution of this work is twofold, (1) it is the first approach of cutting stock problem with multiples objectives that involves the cost of the setup and (2) is the first application of a symbiosis relationship among different species in a genetic algorithm.

The method produces good approaches of the Pareto-Front for larger part of the test problems when taking into account the results of the other three methods.

An aspect that can be improved in the proposed method is the computational time. However, the method has great potential for improvement of its computational time with the use of parallel processing since the three associations can be processed in a parallel and asynchronous way. Besides, a fine adjustment of the algorithm can adapt the method for several circumstances.

Acknowledgements: The authors are very grateful to Shunji Umetani for providing the results obtained by his method and the anonymous referees for many helpful comments which resulted in a greatly improved paper. The authors have been partially supported by M.E.C. (Spain), Project MTM2007-063432. The second author is also supported by CNPq (Brazil), Project 307907/2007-4.

References

Allaby M. (1998). Dictionary of Ecology, Oxford University Press, New York.

Bagchi T.P. (1999). Multiobjective Scheduling by Genetic Algorithms. Kluwer Academic Publishers.

Belov G., Scheithauer G. (2003). The Number of Setups (Different Patterns) in One-Dimensional Stock Cutting, Technical Report MATH-NM-15-2003, Dresden University.

Bremermann H. J. (1962). Optimization through evolution and recombination, In: Self-Organizing Systems [edited by M.C. Yovits, G.T. Jacobi and Goldstine G.D.], pp: 93-106, Spartan Books.

Chang, M., Ohkura, K., Ueda, K. and Sugiyama, M. (2002). A symbiotic evolutionary for dynamic facility layout problem. In Proceedings of the Evolutionary Computation, 1745-1750.

Coello C.A. (2001). An Updated Survey of GA-Based Multiobjective Optimization Techniques,*ACM Computing Surveys* 32, No. 2, 109-142.

Coello, C.A., Van Veldhuizen, D., and Larnont, G. (2002). Evolutionary Algorithms for solving multi-objective problems. Kluwer Academic Publishers.

Coello C.A. (2004). List of references on evolutionary multiobjective optimization. `http://www.lania.rmx/~ccoello/EMOO/`.

Deb K. (2001). Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley.

Diegel, A. (1988). Cutting paper in Richards Bay: dynamic local and global optimization in the trim problem, Orion 3, 42-55.

Diegel, A., Montocchio, E., Walters, E., Schalkwyk, S. and Naidoo, S. (1996). Setup minimising conditions in the trim loss problem. *European Journal of Operational Research* 95, 631-640.

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

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operation Research* 444, 145-159.

Eguchi, T., Hirasawa, K. and Hu, J. (2003). Symbiotic Evolutional Models in Multiagent Systems. The 2003 Congress on Evolutionary Computation 2, 739-746.

Ehrgott, M. and Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum* 22, 425-460.

Eilon, S. (1960). Optimizing the shearing of steel bars. *Journal of Mechanical Engineering Science* 2, 129-142.

Foester, H. and, G. (2000). Pattern Reduction in One-dimensional Cutting-Stock Problems. *International Journal of Prod. Res.* 38, 1657-1676.

Fonseca C.M., Flenfing P.J., Zitzler E., Deb K. and Thiele L. (2003). Evolutionary Multi-Criterion Optimization. EMO 2003, Second International Conference, Faro, Portugal, April 2003 Proceedings, Lecture Notes in Computer Sciences 2632. Springer Verlag.

Fraser, A. S. (1957). Simulation of genetic systems by automatic digital computers: I. Introduction. *Austral. J. Biol. Sci.* 10, 484-491.

Gau, T. and Wascher, G. (1995). CUTGEN1: A Problem Generator for the Standard One-dimensional Cutting Stock Problem, *European Journal of Operational Research 84, 572-579.*

Gilmore, P. C. and Gomory, R. E. (1961). A Linear Programming Approach to the Cutting Stock Problem. *Operations Research* 9, 849-859.

Gilmore, P. C. and Gomory, R. E. (1963). A Linear Programming Approach to the Cutting Stock Problem. *Operations Research* 11, 863-888.

Haessler, R. (1975). Controlling Cutting Pattern Changes in One-Dimensional Trim Problems. *Operations Research* 23, 483-493.

Hardley, C. J. (1976). Optimal cutting of zinc-coated steel strip. *Operational Research* 4, 92-100.

Hinxman, A. (1980). The trim loss and assortment problems: a survey. *European Journal of Operational Research* 5, 8-18.

Hirasawa, K., Ishikawa, I., Hu, J., Jin, C. and Murata J. (2000). Genetic Symbiosis Algorithm, Proceedings of the Congress on Evolutionary Computation 2, pp: 02-xxvi.

Holland, J. H. (1962). Outline for a logical theory of adaptive systems. *J. Assoc. Comput. Mach.* 3,

297-314.
Holland, J. H. (1975). Adaptation in Natural and Artificial Systems, University of Michigan Press.
Johnston, R. E. (1986). Rounding algorithms for cutting stock problems. *Asia-Pacific Journal of Operational Research* 3, 166-171.
Kantorovich, L. V. (1960). Mathematical Methods of Organizing and Planning Production. *Management Sci.* 6, 366-422.
Khalifa, Y., Salem, O. and Shahin, A. (2006). Cutting Stock Waste Reduction Using Genetic Algorithms. Proceedings of the 8th Conference on Genetic and evolutionary computation, 1675-1680.
Kolen, A. W. J. and Spieksma, F. C. R. (2000). Solving a bi-criterion cutting stock problem with open-ended demand: a case study. *Journal of the Operational Research Society* 51, 1238-1247.
Lee, J. (2007). In situ column generation for a cutting-stock problem. *Computers & Operations Research* 34, Issue 8, 2345-2358.
Kim, Y. K., Kim, J. Y. and Kim, Y. (2000). A coevolutionary algorithm for balancing and sequecing in mixed model assembly lines. *Applied Intelligence* 13, 247-258.
Kim, J. Y., Kim Y. and Kim Y. K. (2001). An endosymbiotic evolutionary algorithm for optimization. *Applied Intelligence* 15, 117-130.
Kim, Y. K., Kim, J. Y. and Kim, Y. (2006). An endosymbiotic evolutionary algorithm for the integration of balancing and sequecing in mixed-model U-lines. *European Jounal of Operational Research* 168, 838-852.
Liang, K., Yao, X., Newton, C. and Hoffman, D. (2002). Evaluation of algorithms for one-dimensional cutting. *Computers & Operations Research* 29, 1207-1220.
Mao, J., Hirasawa, K., Hu, J. and Murata, J. (2000). Genetic Symbiosis Algorithm for Multiobjective Optimization Problem, Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication, 137-142.
Margulis, L. (1981). Symbiosis in Cell Evolution, W.H. Freeman, San Francisco.
Metzger, R. W. (1958). Stock Slitting. Elementary Mathematical Programming, Wiley.
Osyczka A. (2001). Evolutionary Algorithms for Single and Multicriteria Design Optimization, Studies in Fuzziness and Soft Computing 79. Physica Verlag.
Paull, A. E. and Walter, J. R. (1954). The trim problem: an application of linear programming to the manufacture of news-print paper. Presented at Annual Meeting of Econometric Society, Montreal, 10-13.
Pianka, E. R. (1994). Evolutionary Ecology. Harper Collins College Publisher, New York.
Potter, M. A. (1997). The design and analysis of a computational model of cooperative coevolution. Ph.D. Dissertation, George Mason University.
Romero, C. (1993). Teoría de la decisión multicriterio: Conceptos, técnicas y aplicaciones, Alianza Editorial.
Moretti, A. C., Salles Neto, L. L. (2008). Nonlinear cutting stock problem model to minimize the number of different patterns and objects. *Computational & Applied Mathematics* 27, 61-78.
Stadler, H. (1990). A one-dimensional cutting stock problem in the aluminum industry and its solution. *European Journal of Operational Research* 44, 209-223.
Steuer, R. E. (1986). Multiple Criteria Optimization: Theory, Computation and Application, John Wiley & Sons.
Tsujimura, Y., Mafune, Y. and Mitsuo, G. (2001). Effects of symbiotic evolution in genetic algorithms for job-shop scheduling, IEEE. Umetani, S., Yagiura, M. and Ibaraki, T. (2003). One Dimensional Cutting Stock Problem to Minimize the Number of Different Patterns. *European Journal of Operational Research* 146, No.2, 388-402.
Umetani S., Yagiura M., Ibaraki, T. (2006). One Dimensional Cutting Stock Problem with a Given Number of Setups: A Hybrid Approach of Metaheuristics and Linear Programming, *Journal of Mathematical Modelling and Algorithms* 5, 43-64.
Umetani S. (2007). `http://www-sys.ist.osaka-u.ac.jp/~umetani/index-e.html`.
Vanderbeck F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems, *Math. Program.* 86, 565-594.

Vanderbeck F. (2000). Exact Algorithm for Minimising the Number of Setups in the One-Dimensional Cutting Stock Problem, *Oper. Res.* 48, 915-926.

Wascher, G., Carow, P., & Muller, H. (1985). Entwicklung eines flexiben Verfahrens für Zuschneideprobleme in einem Kaltwalzwerk. *Zeitschrif für Operations Research* 29, B209-B230.

Wascher, G., (1990). An LP-based approach to cutting stock problems with multiple objectives. *European Journal of Operational Research* 44, 175-184.

Wascher, G. and Gau, T. (1996). Heuristics for the Integer One-dimensional Cutting Stock Problem: a computational study. *OR Spektrum* 18, 131-144.

Wascher, G., Haussner H. and Schumann. (2007). An improved typology of cutting and packing problem. *European Journal of Operational Research* 183, 1109-1130.

Watson, R. A. and Pollack, J. B. (1999). How Symbiosis Can Guide Evolution, Advances in Artificial Life: 5th European Conference, Springer.

Yanasse H.I., Limeira M. (2006). A hybrid heuristic to reduce the number of different patterns in cutting stock problems, *Comput. Oper. Res.* 33, 2744-2756

Zitzler, E., Laumanns, M., and Thiele. M. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm, Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich.

Zitzler, E., Deb K., Thiele L., Coello C. and Corne D. (2001). Evolutionaw Multi-Criterion Optimization, Lecture Notes in Computer Sciences, 1993. Springer Verlag.

Instance	Setups	ILS	crawla	Symbio
Fiber06	7	5.19		
	6	5.19		
	5	5.19		5.19
	4	8.28	8.28	
	3	8.28	12.47	8.28
	2		48.5	17.55
Fiber07	10	4.98		
	9	4.98		
	8	4.98		
	7	4.98		
	6	8.16		
	5			
	4			
	3		8.16	4.98
	2		11.34	11.34
	1		68.61	68.61
Fiber08	10	4.47		
	9	4.47		
	8	4.47		
	7	4.47		
	6	4.47		
	5			
	4			
	3		4.46	4.46
	2		5.67	5.67
	1			126.94
Fiber09	9	9.27		
	8	9.27		
	7	11.29		
	6	11.29		
	5	11.29		
	4		11.29	
	3		23.43	11.29
	2		47.71	47.71
Fiber10	14	4.20		
	13	4.20		
	12	4.20		
	11	4.20		
	10	4.20		
	5		4.20	
	4		5.69	4.20
	3		7.18	5.69
	2		22.06	22.06

Table 6: Setups and percentage trim losses obtained by ILS, crawla and Symbio (5180).

Instance	Setups	ILS	crawla	Symbio
Fiber11	12	6.06		
	11	6.06		
	10	4.52		
	9	4.52		
	8	6.06		
	7			
	6			
	5		6.06	
	4		7.59	6.65
	3		10.67	9.13
	2			52.16
Fiber13a	14	2.41		
	13	4.24		
	12	2.41		
	11	4.24		
	10	4.24		
	8			
	7			
	6			
	5			6.07
	4		13.38	7.90
	3		28.01	13.38
	2			51.79
Fiber13b	9	6.59		
	8	6.59		
	7	6.59		
	6	6.59		
	5	6.59		
	4		6.59	6.59
	3		10.27	10.27
	2		105.83	21.29
Fiber14	11	5.45		
	10	5.45		
	9	5.45		
	8	7.61		
	7	9.76		
	5		5.45	5.45
	4		9.76	9.76
	3		14.06	14.06
	2		63.56	85.08
Fiber15	14	4.76		
	13	4.76		
	12	4.76		
	11	4.76		
	10	6.56		
	4		4.76	4.76
	3		10.17	6.56
	2		49.91	39.07

Table 7: Setups and percentage trim losses obtained by ILS, crawla and Symbio (5180).

Instance	Setups (n)	ILS	crawla	Symbio
Fiber16	10	7.68		
	9	6.46		
	8	8.91		
	7	8.91	4.01	4.01
	6	27.26	5.24	5.23
	5		11.35	7.68
	4		13.80	16.25
	3		45.62	66.41
Fiber17	9	5.46		
	8	6.69		4.24
	7	7.92		
	6	17.28	4.24	5.46
	5	22.63	6.69	6.69
	4		7.92	11.59
	3		29.99	29.99
Fiber18	11	5.10		
	10	6.16		
	9	5.10		5.10
	8	10.41		6.16
	7	11.47		
	6		5.10	7.22
	5		6.16	10.40
	4		7.22	11.46
	3		18.90	82.59
Fiber19	25	4.98		
	24	5.77		
	23	4.98		
	22	4.98		
	21	4.98		
	9		5.77	5.77
	8		6.56	6.56
	7		7.35	7.35
	6		8.14	8.93
	5		9.72	10.51
	4		14.45	24.71
	3			46.82
Fiber20	8	13.41		
	7	19.71	7.11	7.11
	6	26.02	10.26	13.41
	5	54.37	22.86	16.56
	4	57.52	54.37	

Table 8: Setups and percentage trim losses obtained by ILS, crawla and Symbio (5180).

Instance	Setups	ILS	crawla	Symbio
Fiber23	15	4.98		
	14	2.84		
	13	6.41		
	12	9.27		
	11	17.84		8.55
	10			9.26
	9		2.84	10.69
	8		4.27	12.12
	7		5.70	14.98
	6		7.84	17.83
	5		19.98	21.40
	4			67.11
Fiber26	29	3.39		
	28	2.31		
	27	3.39		
	26	3.39		
	25	3.93		
	10			8.78
	9			9.31
	8		4.47	10.93
	7		5.01	15.78
	6		7.16	
	5		15.78	28.16
	4		78.24	96.01
Fiber28a	11	9.70		
	10	10.91		8.49
	9	10.91		9.70
	8	18.14	4.88	13.31
	7	26.58	7.29	19.34
	6		9.70	32.60
	5		13.32	50.69
	4		28.99	
Fiber28b	17			8.27
	15	5.69		
	14	6.55		7.40
	13	6.55		9.12
	12	8.27		
	11	6.55		9.98
	10			10.84
	9		5.69	13.42
	8		5.69	17.72
	7		8.27	22.87
	6		15.14	41.78
	5		34.90	96.77
Fiber29	13	11.03		
	12	9.40		
	11	14.30		7.76
	10	9.40		9.40
	9	12.66	4.50	11.03
	8		9.40	14.29
	7		14.29	22.46
	6		24.09	43.68
	5		32.26	53.48

Table 9: Setups and percentage trim losses obtained by ILS, crawla and Symbio (5180).

Instance	Setups (n)	ILS	crawla	Symbio
Fiber06	5	8.46		
	4	8.46		
	3	8.46	8.46	3.03
	2		19.30	8.46
	1		73.53	73.53
Fiber07	4	5.95		
	3	5.95		
	2	5.95	5.95	5.95
	1		17.10	17.10
Fiber08	4	3.13		
	3	7.34	1.03	1.03
	2	17.87	7.34	3.13
	1		32.60	32.60
Fiber09	5	9.95		
	4	9.95	6.41	
	3	9.95	9.95	6.41
	2		27.69	13.50
	1			254.69
Fiber10	5	6.98		
	4	6.98	1.76	
	3	9.59	4.37	4.37
	2		14.81	9.59
	1			72.21

Table 10: Setups and percentage trim losses obtained by ILS, crawla and Symbio (9080).

Instance	Setups	ILS	crawla	Symbio
Fiber11	5	7.77		
	4	5.08	7.77	
	3	10.47	10.47	5.08
	2		34.71	13.16
	1			228.70
Fiber13a	6	5.79		
	5	8.99		
	4	12.20	8.99	
	3		15.40	8.99
	2		31.43	12.20
	1			406.49
Fiber13b	4	15.97		
	3	9.53		3.08
	2	22.42	9.53	9.53
	1		138.39	138.39
Fiber14	5	5.63		
	4	5.63		1.85
	3	43.35	5.63	5.63
	2		24.49	20.71
Fiber15	5	10.81		
	4	7.64	4.48	
	3	32.97	7.64	4.48
	2		23.47	10.81
	1			200.77

Table 11: Setups and percentage trim losses obtained by ILS, crawla and Symbio (9080).

Instance	Setups	ILS	crawla	Symbio
Fiber16	7			2.96
	6	7.25		5.10
	5	7.25	7.25	7.25
	4	13.68	9.39	9.39
	3		17.97	20.12
	2		45.86	
Fiber17	12	5.33		
	11	5.33		
	10	3.18		
	5		3.18	3.18
	4		5.33	5.33
	3		7.48	13.93
	2		26.83	24.68
Fiber18	6	6.07		
	5	15.37	2.35	2.35
	4	19.09	4.21	4.21
	3		7.93	7.93
	2		33.98	35.84
Fiber19	8	3.77		
	7	6.54		
	6	6.54		3.77
	5			5.15
	4		3.27	6.54
	3		6.10	12.07
	2		49.96	
Fiber20	7	15.97		
	6	15.97		
	5	15.97	10.45	27.12
	4		15.97	37.71
	3		27.01	64.20
	2		137.46	

Table 12: Setups and percentage trim losses obtained by ILS, crawla and Symbio (9080).

Instance	Setups	ILS	crawla	Symbio
Fiber23	7	11.41		
	6	12.67	6.41	8.91
	5	16.42	7.66	10.16
	4		11.41	20.18
	3		20.18	36.45
Fiber26	10			5.72
	9	6.66		6.66
	8	7.61		7.61
	7	8.55		9.49
	6		1.94	10.44
	5		3.83	11.38
	4		10.44	23.65
	3		33.09	46.30
Fiber28a	8	9.88		
	7	9.88		7.77
	6	18.33		9.88
	5		5.66	14.11
	4		11.99	20.45
	3		20.45	41.56
	2		60.60	
Fiber28b	12	3.93		
	11	6.94		
	10	14.47		
	8			9.95
	6		3.93	12.96
	5		6.94	22.00
	4		14.47	41.58
	3		29.53	
Fiber29	13	5.90		
	12	8.76		
	11	11.62		
	9			8.76
	6		5.90	11.62
	5		8.76	17.34
	4		20.21	31.65
	3		45.97	

Table 13: Setups and percentage trim losses obtained by ILS, crawla and Symbio (9080).

Instance	Setups	ILS	Symbio
cutgen01	4	13	
	3	14	
	3		13
	2	15	
cutgen02	8		128
	7		129
	5	123	130
	4	126	134
	3	129	150
	2	139	
cutgen03	6		25
	5		26
	4	24	27
	3	25	30
cutgen04	15	236	
	14	237	
	13	238	
	12	239	
	11	226	
	10	243	
	9	246	
	8	227	252
	7	228	256
	6	231	258
	5	232	264
	4	238	293
	3	249	301
cutgen07	7		80
	5	66	
cutgen08	5	647	647
cutgen09	15	91	96
	14	92	
	13	93	97
	11	94	98
	10	97	99
cutgen10	15	917	979
	14	921	
	13	935	984
	12	940	994
	11	948	1009
	10	970	1059
	9	1065	
cutgen13	7	80	80
cutgen15	13	127	127

Table 14: Setups and number of processed objects obtained by ILS and Symbio in 10 instances of CUTGEN1.