A Comparison between Genetic Algorithms and Evolutionary Programming based on Cutting Stock Problem

Raymond Chiong, *Member, IEEE*, and Ooi Koon Beng, *Member, IEAust*

***Abstract*—Genetic Algorithms (GA) and Evolutionary Programming (EP) are two well-known optimization methods that belong to the class of Evolutionary Algorithms (EA). Both methods have generally been recognized to have successfully solved many problems in recent years, especially with respect to engineering and industrial problems. Even though they are two different types of EA, the two methods share a lot of commonalities in the genetic operators they use and the way they mimic natural evolution. This paper aims to bring forth an introductory review on how these two methods tackle the one-dimensional Cutting Stock Problem (CSP). We draw comparison on the effectiveness of GA and EP in solving CSP, and propose an improved algorithm using a combination of the two methods based on our observations. In the concluding remarks, some future works are suggested for further investigations.**

***Index Terms*—Cutting stock problem, evolutionary programming, genetic algorithms, optimization methods.**

I. Introduction

Many problems in industrial engineering nowadays concern themselves with the goal of an "optimal" solution. Various optimization methods have therefore emerged, being researched and applied extensively to different optimization problems. For small scale problems, exact solution methods such as linear programming can be used effectively. When the problems are large and complex, however, heuristic methods have to be called into play due to the exponential growth of the search space and the time required to find optimal or near optimal solution. Over the past few decades, researchers have proposed many novel nature inspired heuristic methods such as the evolutionary algorithms (EA) for optimization design based on specific domain knowledge. Two well-developed methods belong to the class of EA are Genetic Algorithms (GA) and Evolutionary Programming (EP). In this paper, we present a study to illustrate the use of both GA and EP in tackling the Cutting Stock Problem (CSP), and draw comparison on their effectiveness in finding the optimal solution for CSP.

As one of the classical optimization problems in Operational Research, there are many reasons for CSP to be an interesting topic of research. Its applicability in many industries such as the steel, glass, wood, plastic and paper manufacturing has caused CSP to be widely studied [1-2]. Besides that, CSP also seems to have shared similar structure with some other industrial problems like capital budgeting, processor scheduling, VLSI design, etc. [2]. The capacity of CSP in reflecting the diversity and complexity of the real world problems have definitely intensified the search for better heuristic solutions for it.

The rest of this paper is organized as follows. Section II introduces the background theory of CSP, and describes some existing solution methods for it. Following which, section III and IV show introductory reviews on how CSP is being tackled by GA and EP based on [3] and [4] respectively. We then bring forth some discussion by comparing the effectiveness of both methods on CSP, and present an idea to improve the algorithms in section V. Finally, section VI concludes with a summary of this work together with some possible future works.

II. Cutting Stock Problem

The CSP is a combinatorial optimization problem that involves cutting larger stock sheets into smaller pieces. Generally, two problems arise when small items are to be cut from large objects, which Hinxman [5] called the assortment problem and the trim loss problem. The assortment problem deals with issue of choosing proper dimensions from large objects, whereas trim loss problem addresses the issue of how to minimize wastage in cutting out the smaller items from larger objects. To understand it better, Fig. 1 below shows a simple illustration for the CSP.

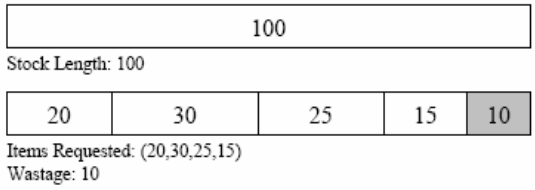

Fig.1. A simple illustration for CSP

From Fig. 1, we see that a stock sheet with fixed length of 100 is available. Given a set of requested items, each with certain length at 20, 30, 25 and 15 to be cut from the stock sheet, the wastage or trim loss will be 10. Here, the stock sheet is referred to as the large object, and the set of requests is the so-called small items. If there is more than one stock sheet to be cut to fulfill several sets of requests, the problem becomes more

Manuscript received March 3, 2006.

Raymond Chiong is with the School of Information Technology, Swinburne University of Technology (Sarawak Campus), State Complex, 93576 Kuching, Sarawak, Malaysia (phone: +60 82 416353 ext 607; fax: +60 82 423594; e-mail: rchiong@swinburne.edu.my).

Ooi Koon Beng is with the School of Engineering, Swinburne University of Technology (Sarawak Campus), State Complex, 93576 Kuching, Sarawak, Malaysia (e-mail: kooi@swinburne.edu.my).

complicated. The purpose of CSP is thus to find cutting patterns that could minimize the number of stock sheets used, and at the same time minimize the wastage resulting from the trim loss.

Many solutions for CSP have been presented since 1960s, and one of the pioneering solutions can be found in the work of Gilmore and Gomory based on a linear programming (LP) approach [6]. Gilmore and Gomory used an LP model with delayed pattern generation technique to solve the trim loss problem efficiently without needing to enumerate all the cutting patterns. Their technique is especially useful when some extremely large stock sheets with millions of possible patterns need to be cut. Besides that, the delayed pattern generation technique is also potential of solving the trim loss problem in a much shorter time [7].

The LP approach of Gilmore and Gomory has subsequently been adopted by many other researchers for various LP-based solutions (see [1] and [5] for more details), and most of them have proven to be successful for CSP along the years. The limitation of LP-based approach, however, is that it focuses solely on minimizing trim loss [4]. When it comes to the real world problems which are mostly non-linear, the LP approach seems to be weak. As a result, researchers nowadays tend to show more interest on the innovative heuristic search methods.

To name just a few, some of the renowned heuristic approaches include the best first search, simulated annealing, tabu search, etc. Unlike the LP approach, heuristic approach is capable of dealing with both the assortment problem as well as the trim loss problem effectively. Moreover, heuristic approach is more convenient in a way that it can produce integer solutions without needing to solve the rounding problems which are common in the LP-based solutions. Nevertheless, heuristic methods are normally very problem-dependent, thus specific domain knowledge is required for finding good heuristics.

Since the 1990s, the field of EA has experienced certain degree of exponential growth. This is evidenced by a range of successful applications of EA in diverse fields such as Engineering, Biomedical, Economics, Operational Research, Robotics, Social Sciences, Physics, Chemistry, etc. EA is a generic term used to describe computer-based problem solving methods based on the concept of biological evolution. They are search algorithms that maintain a population of structures and evolve according to rules of selection as well as other genetic operators like crossover and mutation [8]. By incorporating problem-dependent knowledge using natural data structures and problem-sensitive genetic operators [9], EA can be an extremely promising heuristic approach in finding optimal or near optimal solutions to complex combinatorial optimization problem such as CSP within reasonable computational time.

It is necessary to note that EA differ slightly from other heuristic approaches. One distinctive difference is that EA work within a population or a solution set while other heuristic approaches use a single solution for optimization [9]. The clear advantage of EA over other methods here is that they are able to handle a set of solutions and use their inductive nature to converge the solutions towards optimum without knowing the internal rules. However, by working on a solution set the evolution process of EA has the risk of getting stuck at the local optima. In order to avoid this, EA have to explore the whole search space. When the search space is extremely large, the computational time of the search will increase significantly. Some local searching methods therefore need to be incorporated to speed up the convergence of EA's solutions [9].

There are many specific examples of EA, and most of them are similar in nature but differ in the details of their implementation and the problem domains to which they have been applied to. For the sake of brevity, in this paper we will not discuss all but concentrate merely on the use of GA and EP in tackling the classical CSP.

GA and EP are two important optimization methods in EA. Both have been used successfully to solve a great deal of hard combinatorial optimization problems, and one of such is the CSP. As two major classes of EA, the main difference between GA and EP is that GA uses both crossover and mutation, with crossover as the primary search operator, while EP uses only mutation without crossover. The existing works showed that GA has been applied extensively to solve various kinds of CSPs, but the application of EP is relatively few [4]. Nevertheless, most of the literature after the mid 1990s considered EP to be much simpler, faster and more efficient than GA [4, 10].

Before we describe in details how GA and EP are used to solve the CSP, contiguity or pattern sequencing is another important issue that we need to address. As small items need to be cut from large objects according to some specific patterns, the basic idea of contiguity is to ensure the sequence of cutting patterns can minimize the items which are cut partially.

For CSP without contiguity, the sequence of the patterns is not a concern at all since changing the cutting sequence makes no difference to the optimization result. When a list of items is cut continually, however, we need to consider a queue where partially cut items can be stored and be reused until the requested number of items is completely cut. In large-size real world problems, the arrangement for contiguity issue is quite crucial. For that reason, the industry is trying hard to sequence the patterns in a way that the requested items could be cut in continuous patterns as to minimize the handling costs.

III. Genetic Algorithms for Cutting Stock Problem

In this section, we will give an introductory review on how GA is used to tackle the CSP based on [3]. We consider CSP with contiguity and also without contiguity. First, we address the problem representation on CSP using GA, and subsequently describe the search operators, the fitness function and the replacement strategy.

A. Problem Representation

In general, two representations can be found for solving CSP using GA, namely the group-based representation and the order-based representation. A group-based representation refers to a selection of items that will be cut from stock sheet with same stock length, and an encoder is used to map the

groups. An order-based representation, on the other hand, focuses on the order of the items, and a decoder is used to organize the ordered list into a solution. According to [3], the group-based representation is more favourable to the order-based representation for CSP without contiguity, and they are comparable for CSP with contiguity. The main problem with order-based representation is that the crossover operator in GA will find it hard to exploit the ordering information in it. With ordering information encapsulated within groups in the group-based representation, crossover can work much better.

In GA with group-based representation, the chromosome is represented with a number of groups of items to be cut. A first fit algorithm is normally used to group the items from the requested list into groups which are considered as genes. To form the group, a stock length is first chosen at random by the encoder, and then the items to be cut are picked without replacement based on the first fit algorithm. It is necessary to note that each group will be cut only from a single stock length. As the stock length is implied by the group itself, it is not recorded in the chromosome. The smallest stock length from which the group of items can be cut completely is mapped to each group, as showed in Fig. 2 below.

A number of groups of items:

(10) (6, 5, 2) (15) (6, 5) (7, 6) (10, 5)

Stock lengths available:

12, 13, 15

Mapping Solutions

Items	Stock size to be used	Wastage
(10)	12	2
(6, 5, 2)	13	0
(15)	15	0
(6, 5)	12	1
(7, 6)	13	0
(10, 5)	15	0

Fig. 2. Mapping solutions for group-based GA

The distinctiveness of the mapping solutions above is that the number of groups, or genes, is variable [3]. Besides that, the order of the groups and the order of items in each group have no significance. These made the group-based GA very suitable for CSP.

In GA with order-based representation, the chromosome is represented with an ordering list of all the items to be cut. A decoder is used to form groups from the ordering list based on a given stock length. As the information of stock length is hard to identify in order-based GA, the stock length available is being restricted to only one in [3]. Since the ordering of the items is a concern, a next fit algorithm that selects the next item that fits to form groups from the list of items is a good option. If the next item cannot fit into an existing group, next fit algorithm starts a new group using that item. The only problem with next fit algorithm is that it does not work well with the crossover operator. As a result, the first fit algorithm that is used for group-based GA is also used for the order-based GA, and it has been proven to be successful in [11]. Fig. 3 shows the mapping solutions for order-based representation using a single stock length.

An ordering list of items:

(10, 6, 5, 2, 15, 6, 5, 7, 6, 10, 5)

Single stock length available:

15

Mapping Solutions

Items	Stock size to be used	Wastage
(10)	15	5
(6, 5, 2)	15	2
(15)	15	0
(6, 5)	15	4
(7, 6)	15	2
(10, 5)	15	0

Fig. 3. Mapping solutions for order-based GA

From the above illustration on group-based representation and order-based representation, we can see that the group-based is clearly a better choice with less wastage as compared to the order-based. However, order-based representation is essential especially when we need to deal with CSP with contiguity. This is because with contiguity the order in the chromosome becomes significant. As such, Liang *et al*. [4] proposed an intuitive algorithm in EP to solve this order-based problem in GA. EP will be discussed later in section IV.

B. Reproduction Operators

With different representations, different search operators are used for optimizing the solutions. In this section, we describe the reproduction operators for CSP with and without contiguity separately.

For CSP without contiguity, Hinterding and Khan [3] used a grouping crossover and a group mutation with the group-based GA. Their implementation on the crossover and mutation operators was based on [12]. The grouping crossover builds an offspring by combining a segment of one parent into another parent using an insertion point. In other words, the offspring inherits meaningful information from both the parents with the selection of best possible genes the parents have. The offspring is built by first copying the genes from one parent up to the insertion point. Then the genes from the segment in another parent are copied, followed with the genes after the insertion point from the first parent again. The grouping crossover, however, is not straightforward because the genes from parents have to be added to the offspring with some restrictions to avoid duplicate items. As per the group mutation, a number of genes are chosen and deleted from time to time in order to form new genes in the chromosome. The deleted genes are normally those with greater wastage. The idea behind is to get rid of

those bad genes in hope that the new genes produced would provide a better solution. This mutation operator, however, is more time consuming than the traditional simple swap mutation.

For CSP with contiguity, Hinterding and Khan [3] used the uniform grouping crossover with the order-based GA based on [13]. The uniform grouping crossover gives more significance to the ordering information which is essential in CSP with contiguity. This crossover operator uses a template of binary bits generated at random to exchange some items from both parents to an offspring while at the same time maintain the relative order information. As per the mutation, the remove and reinsert mutation operator is used together with the group mutation described earlier for swapping.

It is necessary to highlight again that the crossover used for GA is not a straightforward one, and its implementation is a tricky and complicated task. It also costs a lot of computational time, and immensely degrades the performance of GA in finding an optimal solution for CSP as a whole.

C. Fitness Function

A fitness function is an objective function that quantifies the optimality of a solution. It is therefore crucial for us to describe the fitness function used to evaluate the optimization results of the CSP here. In this section, we present the fitness function for CSP with and without contiguity based on [3].

For the CSP without contiguity, the fitness function contains two terms, with the first to reduce the wastage and the second to encourage solutions with fewer stock lengths that contain wastage. The cost function can be calculated as below:

$$\text{cost} = \frac{1}{n}\left(\sum_{1,n} \sqrt{\frac{waste_i}{stock\_length_i}} + \frac{number\_wasted}{n} \right)$$

where

n = number of groups
$stock\_length_i$ = the stock length that $group_i$ is to be cut
$waste_i$ = $stock\_length_i$ – sum of items in $group_i$
number_wasted = number of stock lengths with wastage

For the CSP with contiguity, the fitness function again contains two terms, with the first to reduce the wastage and the second to maximize the contiguity by minimizing the number of partially cut items. The cost function can be calculated as below:

$$\text{cost} = \frac{1}{10 + n}\left(\sum_{1,n} \sqrt{\frac{waste_i}{stock\_length_i}} + 10\left(\frac{partial\_items}{diff\_lengths} \right)^2 \right)$$

where

n = number of groups
$stock\_length_i$ = the stock length that $group_i$ is to be cut
$waste_i$ = $stock\_length_i$ – sum of items in $group_i$
partial_items = number of partially cut items
diff_lengths = number of different item lengths

D. Replacement Strategy

The GA in [3] used a steady-state GA based on [13]. Tournament selection with a tournament size of 2 is used to give faster and comparable results. As for the parameter setting, the replacement rate is set from 0 to 100%. Poisson distributed random variable is used to determine the number of genes to be mutated in a chromosome. A new chromosome is produced either through crossover or mutation but not both at the same time in order to know the separate effects of them.

IV. Evolutionary Programming for Cutting Stock Problem

Motivated by the fact that the performance of order-based GA is degraded when crossover is used, Liang *et al*. [4] proposed a new EP for CSP with and without contiguity based on the classical EP in [9]. They used only mutation as the reproduction operator, thus their algorithm is considered to be much simpler than GA. In this section, we will give an introductory review on how EP is used to tackle the CSP. Similar to section III, we first address the problem representation on CSP using EP, and subsequently describe the search operators, the fitness function and the replacement strategy.

A. Problem Representation

The EP proposed in [4] is indeed a very simple algorithm that uses an order-based representation. In EP, the chromosome is represented with an ordering list of all items to be cut without any additional parameter for self-adaptation being used. The cutting points for the ordering list are decided using a decoder. According to [4], the principle of their EP is simply to make a cut before the accumulated item length matches any stock length or exceeds the available stock length. An example is given below for better understanding towards the cutting process.

We assume the request for items to be cut is as follows:

- 2 items of length 3
- 2 items of length 4
- 1 items of length 5
- 3 items of length 6

A chromosome that is randomly generated can be represented with an ordering list like this: (5, 4, 6, 3, 3, 4, 6, 6). Given a single stock length of 12, the cutting solutions and the total wastage are showed in Fig. 4 below.

An ordering list of items:

(5, 4, 6, 3, 3, 4, 6, 6)

Single stock length available:

12

Mapping Solutions

Items	Stock size to be used	Wastage
| 5, 4 |	12	3
| 6, 3, 3 |	12	0
| 4, 6 |	12	2
| 6 |	12	6

Fig. 4. Cutting solutions for EP with single stock length

In the situation when multiple stock lengths are available, the cutting process can be more complex. For example, if there are three stock lengths of 12, 15 and 22, the decoder has to compare and decide on the best cutting points based on the three stock lengths with the aim to minimize wastage. Fig. 5 shows the cutting solutions for EP with multiple stock lengths.

An ordering list of items:

(5, 4, 6, 3, 3, 4, 6, 6)

Multiple stock lengths available:

12, 15, 22

Mapping Solutions

Items	Stock size to be used	Wastage
| 5, 4, 6 |	15	0
| 3, 3, 4 |	12	2
| 6, 6 |	12	0

Fig. 5. Cutting solutions for EP with multiple stock lengths

From the illustration in Fig. 5 above, it is obvious that with the availability of multiple stock lengths, the wastage is significantly reduced. However, the mechanism of finding the best stock length for cutting can be complicated and needs to be handled with extra care.

B. Reproduction Operators

As mentioned before, the EP works without any crossover operator. This is because Hinterding and Khan [3] have observed that the crossover caused degradation in performance of GA, especially for CSP with contiguity. As a result, only a simple swap mutation is used by Liang *et al.* [4] in their proposed EP. It is necessary to note that their simple swap mutation is different from the traditional mutation operator used in classical EP. The reason being that in CSP, the items in the request list are unchangeable. What can be changed or mutated then is only the order of the items.

The simple swap mutation operator works based on a three point swaps (3PS) which swaps three items in a list. For example, the first item is swapped with the second item, and the new first item, which is originally the second item, is then swapped with the third item in the list. The swapping of three items in an ordering list may accelerate the convergence towards the global optimum, if the original list is far away from the global optimum. However, it may also hinder the convergence if the global optimum is already very close to the original list. A balance therefore needs to be maintained in the number of times the 3PS being applied in one mutation.

Apart from the swap mutation operator, a stock remove and insert (SRI) operator is also being incorporated when it comes to CSP with contiguity. SRI is designed for the purpose of rearranging the cutting sequence in order to reduce the number of partially cut items. In SRI, an item is uniformly picked from the ordering list initially. The stock sheet that is used to cut the selected item is then being removed. A search is performed through the ordering list to find another stock sheet that cuts the same length. Once such stock sheet is found, the removed stock sheet is reinserted behind it. Similar to the design of 3PS in simple swap mutation operator, the number of SRI being used within one mutation needs to be devised properly.

C. Fitness Function

For the purpose of comparing the effectiveness of EP with GA, the fitness functions used for CSP with and without contiguity in [4] are exactly the same as the ones used in [3], which have been described in the previous section.

D. Replacement Strategy

The EP used in [4] is a modified one from the classical EP [9]. For CSP without contiguity, tournament selection is used with an initial population being randomly generated. Fitness function is called with the cost value for individual in the population being calculated. Each parent creates a single offspring through mutation. Pairwise comparisons are made over the parents and offspring with opponent size of 10 for each individual. For each comparison, individual with cost lesser than the opponent's survives and will be selected to be the parent for next generation. This replacement strategy is well-known for temporarily keeping some bad genes in order to maintain the diversity throughout the entire search procedure. It is necessary to note that the opponent size used has a direct influence to the speed of convergence. As for CSP with contiguity, the procedure is basically the same with CSP without contiguity apart from that SRI is used together with 3PS as the mutation operators.

V. Comparison and Discussion

The GA and EP that we have reviewed in the previous sections are two of the most significant optimization methods proposed for CSP with and without contiguity to date. From our reviews, we see that the group-based GA has significant advantage over the order-based GA, especially in the case of CSP without contiguity. The crossover operator presents a major problem for the order-based GA, even though the uniform grouping crossover has been proposed for better performance in [3]. In some occasions, the crossover operator used in order-based GA could even destruct the ordering information in the chromosome. We believe that the decoder used for mapping the solutions after crossover takes place could be harmful to the useful grouping information, thus making the crossover nothing more than a random swap.

Realizing the importance of the ordering information and the disadvantage of the crossover in order-based GA, EP has been presented in [4] with the aim of using mutation as the only search operator. A much simpler and less time-consuming algorithm has thus emerged. Based on our observation, we believe that EP outperforms GA because the reproduction operators used in EP for mapping the solutions are much more heuristic and simpler than those used in GA. On the other hand, we have also found that the steady-state replacement strategy used for GA in [3] has a good balance of replacement rate for

better convergence.

Drawing from the comparison made between GA and EP, we believe that a combination of the two methods can produce an improved algorithm for solving CSP. In EP, the SRI is deployed to gather together all the stock sheets that can be used for items with same length. As SRI operates based on groups, this mutation operator is still not able to gather items with the same length into the same stock sheets for more contiguity. In view of this restriction, we propose an order-based SRI mutation for better optimization. The new order-based SRI can be implemented as follows:

1. Select an item uniformly at random from the ordering list.
2. Remove the selected item from the ordering list.
3. Search through the ordering list to find the item that has the same length.
4. Insert the removed item right behind the first such found item.

Our proposed algorithm also incorporates the idea of the steady-state replacement strategy from GA [3] for a better convergence in the replacement process. Experiments need to be conducted to verify the improved algorithm.

VI. Conclusion and Future Work

The CSP has gained a lot of attention as a mean to increase efficiency for many industrial problems nowadays. In this paper, we reviewed and made some general observations on how GA and EP were used to solve the CSP. Several components deemed to be important in the applications of these methods are the problem representation, the search operators, the fitness function and the replacement strategy. Most important of all here is the issue of representation, for without proper and domain-specific representation, we simply cannot apply these EA to solve real world problems. A good replacement strategy is also essential to improve the efficiency of the optimization process.

We have also proposed an improve algorithm based on the combination of GA and EP, using the steady-state replacement strategy and an order-based SRI mutation. We believe that our proposed algorithm can improve the optimization results of CSP significantly. As EA are heuristic in nature, finding good design for the parameter settings is a major task and requires substantial experience. Extensive experiments therefore need to be done to verify our proposed algorithm and make it more effective.

As a matter of fact, there are still a lot of rooms for improvement in using EA to tackle CSP. One of such is to investigate the use of self-adaptation for the evolution process in applying the search operators. Self-adaptation is able to handle the search biases and evolve the replacement rate of the steady-state replacement strategy for better control over the optimization process. Other than that, investigation on better heuristics especially for the mapping process can also be carried out.

Acknowledgment

The authors would like to thank Professor Xin Yao from the School of Computer Science, University of Birmingham, UK for his feedback. The authors would also like to thank Dr Argenes Siburian from the School of Engineering, Swinburne University of Technology (Sarawak Campus) for his helps and comments.

References

[1] H. Dyckhoff, "A typology of cutting and packing problems," *European Journal of Operational Research*, vol. 44, pp. 145-159, 1990.

[2] E. E. Bischoff and G. Wascher, "Cutting and packing," *European Journal of Operational Research*, vol. 84, pp. 503-505, 1995.

[3] R. Hinterding and L. Khan, "Genetic algorithms for cutting stock problems: with and without contiguity," in *Progress in Evolutionary Computation* (X. Yao, ed.), vol. 956 of *Lecture Notes in Artificial Intelligence*, Berlin, pp. 166-186, Springer, 1995.

[4] K. H. Liang, X. Yao, C. Newton, and D. Hoffman, "A new evolutionary approach to cutting stock problems with and without contiguity," *Computers and Operations Research*, vol. 29, pp. 1641-1659, 2002.

[5] A. Hinxman, "The trim-loss and assortment problems: A survey," *European Journal of Operational Research*, vol. 5, pp. 8-18, 1980.

[6] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting stock problem – part II," *Operations Research*, vol. 11, pp. 863-88, 1963.

[7] P. Sweeney and E. Paternoster, "One-dimensional cutting stock decision packing problems," *Journal of the Operational Research Society*, vol. 43, no. 7, pp. 691-706, 1992.

[8] W. M. Spears, K. A. De Jong, T. Back, D. B. Fogel, and H. de Garis, "An overview of evolutionary computation," in *Proceedings of the European Conference on Machine Learning (ECML'93)*, Vienna, Austria: Springer Verlag, 1993, vol. 667, pp. 442-459.

[9] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs: A Hierarchy of Evolution Programs*, Berlin: Springer, 1996.

[10] D. Fogel, "A comparison of evolutionary programming and genetic algorithms on selected constrained optimization problems," *Simulation*, vol. 64, no. 6, pp. 397-404, 1995.

[11] R. Hinterding, "Mapping, order-independent genes and the knapsack problem," in *Proceedings of the First IEEE Conference on Evolutionary Computation (ICEC'94)*, Orlando, Florida, USA, 1994, pp. 13-17.

[12] E. A. Falkenauer and A. Delchambre, "A genetic algorithm for bin packing and line balancing," in *Proceedings of the IEEE International Conference on Robotics and Automation (RA92)*, Nice, France, 1992, pp. 1186-1193.

[13] L. Davis, ed., *Handbook of Genetic Algorithms*, New York: Van Nostrand Reinhold, 1991.