2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Fitness Level based Adaptive Operator Selection for Cutting Stock Problems with Contiguity

Kai Zhang
School of Computer Science and Technology, University of Science and Technology of China(USTC), Hefei, Anhui, China 230007
Email: zkd1989@mail.ustc.edu.cn

Thomas Weise
School of Computer Science and Technology, University of Science and Technology of China(USTC), Hefei, Anhui, China 230007
Email: tweise@ustc.edu.cn

Jinlong Li
School of Computer Science and Technology, University of Science and Technology of China(USTC), Hefei, Anhui, China 230007
Email: jlli@ustc.edu.cn

***Abstract*—In this article, we propose the Fitness Level based Adaptive Operator Selection (FLAOS). In FLAOS, the discovered objective values are divided into intervals, the fitness levels. A probability distribution corresponding to a fitness level describes the selection probabilities of a set of operators. An evolutionary algorithm with FLAOS is suggested to solve one-dimensional cutting stock problems (CSPs) with contiguity. These problems are bi-objective and the goals are to minimize the trim loss and to minimize the number of partially finished items. Experimental studies have been carried out to test the effectiveness of the FLAOS. The solutions found by FLAOS are better than or comparable to those solutions found by previous methods.**

***Keywords*—cutting stock problems, fitness level, AOS.**

I. Introduction

The cutting stock problem (CSP) is a traditional class of combinatorial optimization problems [1]. The goal of a CSP is cutting a set of big objects, made of a material such as reels of wire, paper, steel, plastic, aluminum, or pieces of wood, into a set of small items, to satisfy the customers' demands. The main objective of the CSP is the trim loss or cost minimization [2]. Usually, CSP solution methods are designed for a special objective function [3]. Dyckhoff [4] and Wäscher [5] categorize the different objectives according to the characteristics of CSPs. The applicability of CSP in many industries such as steel, glass, wood, plastic and paper manufacturing has caused them to be an interesting topic for research [6]. Besides that, CSPs have a structure similar to other industrial problems like VLSI design, capital budgeting, etc. [7].

In this study, we propose a new method named Fitness Level based Adaptive Operator Selection [8], [9] (FLAOS) to solve CSPs with contiguity. In the population of an Evolutionary Algorithm (EA) or Evolutionary Programming (EP) process, there are many individuals with different fitness values. An interval of fitness values is called a *fitness level*. For each fitness level, FLAOS defines and adapts a probability distribution for applying different candidate search operators. If the fitness value of an individual belongs to a given fitness level, the operator used to generate its offsprings is selected according to the corresponding probability distribution.

The main contributions of our work are as follows: Firstly, we propose the fitness level and combine it with AOS (FLASO) to solve the CSPs with contiguity; Secondly, experiment studies show that FLAOS could achieve better performance on a set of problem instances from the literature [3], [10] and a set of randomly generated problem instances.

The rest of the paper is organized as follows. Related work is discussed in the next section. Section III presents the description of CSPs with contiguity. Section IV introduces the two operators and the detailed description of FLAOS. In section V experimental studies show that FLAOS could solve the CSPs with contiguity efficiently. Section VI concludes this study and discuss the future work.

II. Related Work

In 1DCSPs, standard-length stocks are cut to produce a number of required small items with a minimal amount of waste. The main objective of classical CSPs is minimizing the trim loss. The most common approaches to solving CSPs are linear programming methods [11], [12] and heuristic methods [13], [14], [15], [16], [17], [18]. Recently, effective heuristic approaches were proposed for the classical CSP:

In [18], Cherri et al. introduced a heuristic procedure which modified the heuristics first-fit-decreasing (FFD) leftover and residual greedy rounding (RGR) leftover [17] to make a priority for the leftovers. The larger leftovers had higher priority for use latter. FFD heuristic strategy considered the use of no-standard stocks only. The RGR heuristics reduced the stock of retails and maintained lower loss solutions.

Besides the objective of trim loss minimization, many other objectives can be defined for CSPs, such as minimizing the total number of stocks cut, maximizing the profit, minimizing production cost due date, etc. The definition of the CSPs vary significantly with the different objectives and so do the algorithms for solving them. In this study, we mainly concern the objectives for ordered demand, which is an important issue in CSPs.

In the current industry, delivering orders on time may be more important than the scrap or the cost of materials [19]. Recently, there are many new works concerning about the sequencing objective in CSPs. Reinertsen et al. [19] consider CSPs with due dates, i.e., those where each order must be completed at a certain time. He applied the column generation

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

approach [15] to initialize the cutting pattern, followed by a heuristic method to find the solutions. For CSPs that minimize the maximum number of open stacks (MOSP), Yanasse et al. [20] used Lagrangian relaxation to decompose the problem into a CSP and a pattern sequencing problem (PSP). The CSP is then solved with the column generation approach and the PSP with a heuristic method. In [21], a genetic algorithms are used to solve the ordered CSPs, in which several items belong to a job and the jobs need be finished in a particular order. For CSPs with different objectives, there are different heuristic approaches, and the effectiveness of approaches is related to the definition of problems.

The objective of contiguity is minimizing the number of partially finished (open) items, which is cutting the same type of items continuously as much as possible [10], [13]. This sequencing factor is crucial in industrial manufacturing. For example, in the wood hard board industry [20], a cutting machine has a limited number of open stacks to store the items until they are used. One stack can only hold one type of item, and it cannot hold a new item until the remaining type of item has been completed. This results from the limitation of physical space or the use of restricted resources to control the stacks.

There are few works dealing with the objective of contiguity, a realistic model of real-world situations. In this study, we consider CSPs on two objectives: trim loss minimization and the number of partially finished items minimization, which is called CSPs with contiguity.

For optimizing CSPs with contiguity, there are two main heuristic methods in use: genetic algorithm (GA) and evolutionary programming (EP). The main difference between GA and EP is the search operators used and the replacement strategy [6]. Hinterding and Khan [10] designed GA with a special fitness objective function that combines two sub-objectives – trim loss minimization and contiguity. In their method, crossover was used as primary search operator. However, the authors found that the crossover operator applied to the order-based representation degrades the performance of GA. Liang et al. [3] proposed EP to solve the CSPs with contiguity. They used the same objective function and introduced three new features. First, for the problem representation, one integer vector holding an ordered list of all requested items is used. Second, no crossover is applied. Third, a three point swap (3PS) mutation is applied. In [3], 20 problem instances were described in detail. Experiments showed that EP outperformed the GA [10] or a heuristic algorithm named two-swap algorithm [3] on the 20 problem instances.

Even though EP [3] was an effective approach for CSPs with contiguity, there are many parameters which need be fine-tuned. For example, the number of 3PS mutation would affect the performance of EP. Performing more 3PS mutations favors exploration whereas fewer 3PS mutations lead to increased exploitation. Liang et al. spent effort on deciding the number of 3PS mutation. They used problem instances 4a and 5a for experimentation and decided that the number of 3PS mutation was two. However, this parameter value might only perform well on these two problem instances.

Finding an appropriate parameter setting for an EA is crucial for good performance [22], [23], [24]. EP has parameters such as the population size, the mutation operator, the mutation rate, etc. [25]. The parameter values greatly affect whether the algorithm can find an optimal result effectively or not [26].

In [3], there is another mutation named stock remove and insert (SRI). The two mutations (3PS and SRI) have significant influences on the performance [3]. In the following, we will focus on controlling these operators, but also discuss the influence of some other parameters, such as the tournament and population sizes.

III. Cutting Stock Problems With Contiguity

In the CSPs with contiguity, there may be several types of stock lengths and the available quantity for each type of stock length is unlimited. If there is only one type of stock length, the problem is called single stock length CSPs, otherwise, it is called multiple stock length CSPs. The goal of the CSPs with contiguity is to produce the required items by cutting the stock lengths to minimize the total waste of material and the number of partial finished items. For an ordered list of items, a cut is made before the accumulated item length matches any stock length. If the accumulated item length exceeds the maximum stock length, a cut is made at the maximum stock length [3].

To formulate the mathematical model for the CSPs with contiguity, we assume the following terminology:

- n: number of types of items;
- K: number of types of stocks;
- m: number of total stocks cut which are actually used to produce requested items;
- l_i: length of item type i, $i = 1, \ldots, n$;
- S_k: length of stock type k, $k = 1, \ldots, K$;
- d_i: number of demand for item type i, $i = 1, \ldots, n$;
- L_j: length of the j^{th} stock cut, $j = 1, \ldots, m$;
- x_{ij}: number of item type i in the j^{th} stock cut;
- w_j: waste of the j^{th} stock cut, given by Function (2);
- y_{ij}: open status of item type i in the j^{th} stock cut, $i = 1, \ldots, n$, $j = 1, \ldots, m$, given by Function (4);
- o_j: number of partially finished (open) orders up to the j^{th} stock cut, $j = 1, \ldots, m$, given by Function (3).

For an ordered list of requested items, a corresponding ordered list of stocks cut is $(L_1, \ldots, L_j, \ldots, L_m)$. This entails an ordered list of wastes $(w_1, \ldots, w_j, \ldots, w_m)$ and the list of the number of partially finished orders up to the j^{th} stock cut $(o_1, \ldots, o_j, \ldots, o_m)$. For the j^{th} stock cut, y_{ij} represents the open status of item type i: If we have begun to cut item type i but have not completed the total demanded d_i of that item type, then $y_{ij} = 1$, otherwise $y_{ij} = 0$. The o_j represents the sum of open status of each item type in the j^{th} stock cut.

The solution of the CSPs with contiguity in our study is an ordered list of total requested items. For example, Figure 1 shows an ordered-based representation $X = (I_1, \ldots, I_8)$ and the mapping from the representation to a solution $X = (5, 4, 6, 3, 3, 4, 6, 6)$. The shadow part of the stock is the waste.

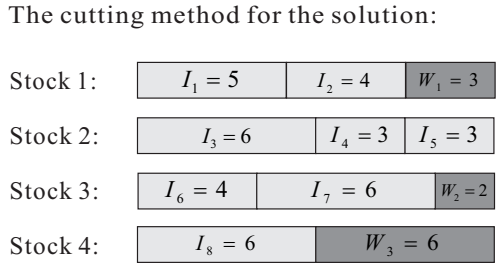

Fig. 1. An example is given as above for better understanding to the cutting process. A permutation of the eight requested items is the solution X. The solution X is an ordered list. Four stocks are used. The shadow part of the stock is the waste, and the waste of second stock is 0. The total waste is 11.

The stock length is $L = 12$. There are $d_1 = 2$ items of length $l_1 = 3$, $d_2 = 2$ items of length $l_2 = 4$, $d_3 = 1$ items of length $l_3 = 5$, and $d_4 = 3$ items of $l_4 = 6$.

We use the same objective function for CSPs with contiguity as [3], [10], a cost function based on a sum of two terms. The first part is minimizing the waste of stocks cut and the second part is minimizing the number of open items.

$$Cost = \frac{1}{m+10}\left(\sum_{j=1}^{m}\sqrt{\frac{w_j}{L_j}} + \frac{10}{m}\sum_{j=1}^{m}\left(\frac{o_j}{n}\right)^2\right) \quad (1)$$

where $Cost$ is a function of w_j and o_j:

$$w_j = L_j - \sum_{i=1}^{n} x_{ij} l_i, \quad i = 1,\ldots,n,\ j = 1,\ldots,m \quad (2)$$

$$o_j = \sum_{i=1}^{n} y_{ij}, \quad i = 1,\ldots,n \quad (3)$$

$$y_{ij} = \begin{cases} 0, & if \sum_{k=1}^{j} x_{ik} = 0 \text{ or } \sum_{k=1}^{j} x_{ik} = d_i \\ 1, & \text{otherwise} \end{cases} \quad (4)$$

subject to

$$\sum_{j=1}^{m} x_{ij} = d_i, \quad i = 1,\ldots,n,\ j = 1,\ldots,m \quad (5)$$

IV. The Fitness Level Based AOS Approach For CSPs With Contiguity

FLAOS has two main components, the fitness level and the operator selection. For each fitness level, operator selection maintains a probability distribution of candidate search operators. We will select search operators according to the probability distribution. Then according to the evaluation of the performance of the selected operators, the corresponding probability distribution of fitness level will be updated. Figure 2 shows the main scheme of FLAOS.

A. Operators For CSPs With Contiguity

There are two primary search operators for CSPs with contiguity, 3PS mutation and SRI mutation [3]. The 3PS mutation operator is mainly minimizing the waste. The SRI mutation operator is mainly minimizing the number of partially finished items. To obtain good results in terms of the $Cost$ function, the usage of the two operators needs to be well balanced.

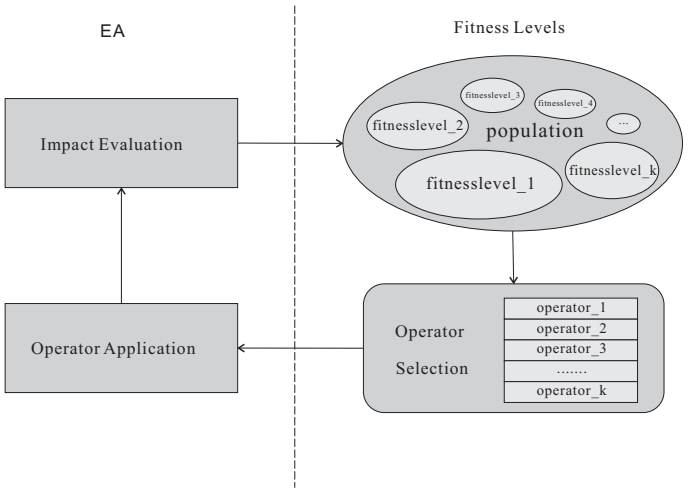

Fig. 2. The scheme of Fitness Level based Adaptive Operator Selection.

3PS mutation will select three items [3]: the first item $x_{ij}^{'}$ is selected uniformly at random from the ordered list of requested items. The second and third item are selected as follows. The stock L_j is selected at random according to the probability given by Function (6), and an item $x_{ij}^{''}$ is selected uniformly randomly from the j^{th} stock.

$$Pr(j) = \frac{\sqrt{\frac{1}{w_j}}}{\sum_{j=1}^{m}\sqrt{\frac{1}{w_j}}}, \forall w_j \neq 0, \quad (6)$$

3PS mutation swaps the first item with the second one, and then swaps the new first one with the third one. For one stock length L_j, there are different cutting patterns, i.e., methods of cutting a stock length into different items [14]. A solution consists of an ordered list of items, which corresponds to an ordered list of stocks cut. 3PS mutation swaps the items among these stocks cuts, which is to select different cutting patterns. According to Function (6), the stocks with more waste are preferred for new cutting patterns.

SRI mutation is designed for considering contiguity [3]. SRI is used for reducing the number of partially finished items through rearranging the cutting sequence: firstly, it selects an item uniformly at random from the candidate solution (ordered list); secondly, it removes the stock that consists of the selected item; thirdly, search through the ordered list of stocks cut to find the stock that cuts the item that has the same length; finally, it inserts the removed stock right behind the first such found stock. SRI mutation makes the same length of items to be cut as closely as possible.

B. Fitness Level And Operator Selection

A fitness level is an interval of fitness values. We define $Level$=$\{level_0, level_1, level_2, \ldots, level_k, \ldots\}$. We denote the $level_k$=$[f_{l_k}, f_{u_k})$, where f_{l_k} is the lower bound and f_{u_k} is the upper bound. Since each individual in the population belongs to a fitness level, we can write $Population_i = pop_i^0 \cup pop_i^1 \cup pop_i^2 \cup \ldots \cup pop_i^k \cup \ldots$. The i represents the i^{th} generation. The fitness value of an individual ind_j (with $j \in 1..\mu$) is presented as f_{ind_j}. We denote that pop_i^k=$\{ind_j | f_{ind_j} \in level_k\}$. The

Algorithm 1: $FLAOS$

```
Initialize fitness level and evaluate the population;
counter ⟵ 0;
while not termination do
    for i ← 1 to μ do
        Prob_{X,Y} ← FitnessLevel(ind_i);
        sample x and y following Prob_{X,Y};
        offspring ← SwapSRIMutate(ind_i, x, y);
        evaluate offspring;
        if f_offspring is minimally better than the lower
        bound of fitness levels then
            counter++;
        end
        if f_offspring is minimally better than f_ind_i then
            UpdateDistribution(Prob_{X,Y});
        end
    end
    Select next generation according to score;
    if counter > 1/3 μ then
        Create a new fitness level;
        counter ⟵ 0;
    end
end
```

individuals of better fitness levels may be closer to a local or global optimum.

Multiple applications of 3PS or SRI mutations usually mean large-step search, and fewer applications usually mean small-step search. Corresponding to each fitness level, there is a joint probability distribution $Prob_{X,Y}$, where X and Y are two random variables. X represents the number of 3PS mutations, and Y represents the number of SRI mutations. When an individual undergoes mutation, it will draw two numbers x and y for mutation operators based on this joint probability distribution $Prob_{X,Y}$.

In FLAOS, fitness improvement is the main evidence used to adapt the joint probability distribution per fitness level. An offspring is created by applying x times 3PS and y times SRI. If the offspring outperforms the parent, we will update the $Prob_{X,Y}$ of the fitness level of the parent. The probability of x times of 3PS mutation and y times of SRI mutations will be increased.

C. Description Of FLAOS

FLAOS is presented in detail in Algorithm 1. We initially generate and evaluate μ individuals randomly. The population size is fixed during the run. The number of initial fitness levels is two. To calculate the number of individuals whose fitness values do not belong any current fitness level, we define a variable $counter$ initialized to 0. The function $FitnessLevel(ind_i)$ returns the fitness level of ind_i. If the fitness value of the individual is better than the current lower bound of fitness levels, we chose the fitness level whose interval is close to the fitness value of the individual, and the $counter$ is increased by 1. According to the joint probability distribution $Prob_{X,Y}$ of the selected fitness level, we generate x and y. The function $SwapSRIMutate(ind_i, x, y)$ creates an offspring of ind_i by applying x times the 3PS and y times the SRI operator. We calculate the fitness value of the offspring. If the offspring outperform its parent, the probability to select the x and y in $Prob_{X,Y}$ of parent will be increased. In total, μ offspring individuals are generated and evaluated this way.

In selection, every individual has a score initialized to 0. For each individual, q opponents are chosen randomly from the $(\mu+\mu)$ population. The number of the q opponents whose fitness are worse than the individual will be the individual's score. Then μ individuals are selected to be the next generation according to the score.

When the $counter$ is greater than one third of μ, we create a new fitness level. This fitness level controls an interval that spans from the lower bound of origin fitness levels to the best fitness value of current population. Then the $counter$ is reset to 0.

TABLE I. FEATURES OF ALL PROBLEMS INSTANCES.

inst	total number of requested items	number of stock types	number of item types	inst	total number of requested items	number of stock types	number of item types
1	20	3	8	1a	20	1	8
2	50	3	8	2a	50	1	8
3	60	6	8	3a	60	1	8
4	60	3	8	4a	60	1	8
5	126	8	18	5a	126	1	18
6	200	5	18	6a	200	1	18
7	200	5	24	7a	200	1	24
8	400	3	24	8a	400	1	24
9	400	6	36	9a	400	1	36
10	600	3	36	10a	600	1	36
11	800	8	10	11a	800	1	10
12	800	8	40	12a	800	1	40
13	800	8	80	13a	800	1	80
14	1800	8	10	14a	1800	1	10
15	1800	8	40	15a	1800	1	40
16	1800	8	80	16a	1800	1	80
17	3000	8	10	17a	3000	1	10
18	3000	8	40	18a	3000	1	40
19	3000	8	80	19a	3000	1	80

V. EXPERIMENTAL STUDY

In order to evaluate the performance of FLAOS, we compare it with EP [3] on the instances from the literature [3], [10]. To further evaluate the effectiveness of the fitness level based adaptation, we compared our approach to an EA that uses a single operator probability distribution for the whole population. The results of two objectives, waste minimization and contiguity were also presented in detail for each algorithm. We additionally apply the algorithms to some new random problem instances.

A. Experimental Settings And Description Of Problem Instances

There are 38 problem instances ready to test. The first 10 benchmark problem instances (1-5 and 1a-5a) were from Hinterding and Khan [10]. The detailed description of the other 10 problem instances (6-10 and 6a-10a) were given in [3]. The last 18 problem instances 11-19 and 11a-19a are randomly generated. Problem instances 1-19 are multiple stock length CSPs and problem instances 1a-19a are single stock length CSPs. The total number of requested items increases as the problem index increases.

In Table I, we present the features of all problem instances in detail. The new problem instances 11-19 and 11a-19a are generated deliberately. We consider three features for the new problem instances: the total number of requested items, the number of stock types and the number of item types. These three features are given small or big values. Firstly, the total number of requested items is much larger than the biggest total number (600) of requested items of problem instances in [3]. Secondly, the number of item types has a larger range (from 10 to 80) than the first twenty instances (8 to 36). Thirdly, the numbers of stock types of the new instances have two different values, either 1 or 8.

In the following experiments, the population size μ is set to 75. The tournament size q is set to 10 in selection. These parameters are the same as used in EP [3]. Each problem instance is repeated for 50 independent runs. The ranges of x and y in our experiments are from 0 to 8.

B. Results Of Problem Instances From Literature

In Table II, *inst* is the index of problem instance. *inum* is the total number of requested items. *gen* is the number of generations. *Mean* is the mean best fitness. *Std Dev* is the standard deviation. For the results of each row, the bold result with an underline is the best result, the bold result without an underline is the better one, and the result without bold is the worse one.

The results in Table II show that FLOAS has performed better than EP for most problem instances. AOSEP performs at least as well as the EP for ten benchmark problems. FLAOS performs better than EP for 12 problem instances which are instances 2-5, 8, 10 and 3a-4a, 7a-10a. The results of FLAOS is comparable to EP's for instances 1, 6, 1a, 5a and 6a. The results of FLAOS are worse than EP's on problem instances 7 and 9. The standard deviations of problem instances 1-3, 6-8, 10, 1a-4a and 7a-10a are smaller than EP's, which indicates that our solutions are more robust and reliable. In Table II, we also present the results of AOSEP. AOSEP outperforms EP on six problem instances 2-4, 3a, 7a and 10a. The results is comparable to EP on nine problem instances 1, 5, 1a, 2a, 4a, 8, 10, 8a, 9a. We can conclude that the results of FLAOS is better than EP for most problem instances from [3], [10].

To better investigate the influence of FLAOS on the two objectives (waste minimization and contiguity), we present the detailed results for the three algorithms EP [3], FLAOS, and AOSEP simultaneously in Table III.

In Table III, *Stock used* indicates the average number of stock used, *Total wastage* is the average total wastage from all used stock, *Stocks w/wastage* is the average number of stocks with wastage, and *Max open* is the average maximum number of open items in the solution. For the problem instances 1-5 and 1a-5a [10], the difference between FLAOS and EP is very small on the total wastage and the number of stocks with wastage, but FLAOS outperform EP on the maximum number of open items. FLAOS cuts stocks without any waste for six problem instances, while EP achieves this only for five. The results of total wastage and number of stocks with wastage of FLAOS is less than EP's on problem instances 8, 10, 7a-10a. The results of number of stocks used of FLAOS is better than EP on problem 8, 7a-10a. The results of maximum number of open items of FLAOS are worse than the results of EP on problem instances 8, 10, 6a-10a. This is due to the number of consecutive SRI mutations used of EP is 4 [3], which is much more than the number of SRI mutations of FLAOS.

TABLE II. THE MEAN BEST RESULTS OF THE EP, THE FLAOS AND THE AOSEP FOR PROBLEM 1-10 AND 1A-10A OVER 50 INDEPENDENT RUNS.

inst	inum	gen	EP		FLAOS		AOSEP	
			Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
1	20	500	9.36e-3	1.17e-3	**9.06e-3**	**8.46e-4**	**9.01e-3**	**9.51e-4**
2	50	1000	1.41e-2	3.42e-3	**1.22e-2**	**2.45e-3**	**1.21e-2**	**2.21e-3**
3	60	2000	1.66e-2	3.42e-3	**1.12e-2**	**1.23e-3**	**1.15e-2**	**1.79e-3**
4	60	2000	2.61e-2	4.88e-3	**1.97e-2**	**5.54e-3**	**2.11e-2**	**5.90e-3**
5	126	2000	**7.28e-3**	**1.50e-3**	**6.81e-3**	**1.67e-3**	7.41e-3	1.54e-3
1a	20	500	4.40e-2	0	4.40e-2	0	4.40e-2	0
2a	50	1000	**6.83e-2**	**3.64e-3**	**6.81e-2**	**3.35e-3**	6.95e-2	4.41e-3
3a	60	2000	6.26e-2	1.63e-2	**5.13e-2**	**1.20e-2**	**5.15e-2**	**1.05e-2**
4a	60	2000	**6.57e-2**	**8.19e-3**	**6.39e-2**	**6.93e-3**	6.62e-2	7.67e-3
5a	126	2000	**1.25e-1**	**4.92e-3**	**1.29e-1**	**7.19e-3**	1.31e-1	6.35e-3
6	200	3000	**1.39e-2**	**6.20e-3**	**1.41e-2**	**5.57e-3**	1.76e-2	6.36e-3
7	200	3000	**1.55e-2**	**3.02e-3**	**1.64e-2**	**1.91e-3**	1.70e-2	2.11e-3
8	400	5000	**3.24e-2**	**6.89e-3**	**2.23e-2**	**5.07e-3**	3.33e-2	1.19e-3
9	400	5000	**1.04e-1**	**2.31e-3**	**1.21e-1**	**2.52e-3**	1.42e-2	3.85e-3
10	600	10000	**3.62e-2**	**5.31e-3**	**3.22e-2**	**4.67e-3**	3.83e-2	6.55e-3
6a	200	5000	**1.05e-1**	**1.02e-2**	**1.08e-1**	**9.22e-3**	1.13e-1	1.22e-2
7a	200	5000	7.14e-2	1.11e-2	**5.56e-2**	**1.18e-2**	**6.64e-1**	**1.46e-2**
8a	400	5000	**1.34e-1**	**9.13e-3**	**1.20e-1**	**8.70e-3**	1.38e-1	1.30e-2
9a	400	10000	**7.89e-2**	**7.76e-3**	**6.96e-2**	**7.80e-2**	8.12e-2	1.46e-2
10a	600	20000	8.44e-2	7.06e-3	**6.41e-2**	**6.22e-2**	**7.84e-2**	**1.23e-2**

TABLE IV. THE MEAN BEST RESULTS OF FLAOS AND AOSEP FOR THE NEW PROBLEM INSTANCES OVER 50 INDEPENDENT RUNS.

inst	inum	gen	FLAOS		AOSEP	
			Mean	Std Dev	Mean	Std Dev
11	800	10000	**3.29e-2**	**4.17e-3**	3.71e-2	4.36e-3
12	800	10000	**2.03e-2**	**8.81e-4**	2.25e-1	2.29e-3
13	800	10000	**1.21e-2**	**1.62e-3**	1.65e-2	6.81e-3
14	1800	10000	8.87e-2	2.01e-3	**8.75e-2**	**3.48e-3**
15	1800	10000	**6.03e-2**	**2.40e-3**	6.91e-1	4.01e-3
16	1800	10000	**5.60e-2**	**1.94e-3**	6.53e-1	2.85e-3
17	3000	10000	**5.42e-2**	**2.70e-3**	6.33e-1	4.26e-3
18	3000	10000	7.39e-2	2.86e-3	**6.87e-2**	**4.02e-3**
19	3000	10000	**7.13e-2**	**2.61e-4**	8.02e-2	5.91e-3
11a	800	10000	**1.44e-1**	**7.29e-3**	1.61e-1	1.04e-2
12a	800	10000	**1.07e-1**	**8.01e-3**	1.33e-1	1.29e-2
13a	800	10000	**1.31e-2**	**9.36e-3**	1.47e-1	1.84e-2
14a	1800	10000	**2.13e-1**	**3.47e-3**	2.20e-1	8.82e-3
15a	1800	10000	**1.98e-1**	**2.35e-3**	2.11e-1	9.47e-3
16a	1800	10000	**2.00e-1**	**1.55e-3**	2.12e-1	3.56e-2
17a	3000	10000	**2.21e-1**	**3.00e-3**	2.27e-1	9.30e-3
18a	3000	10000	**2.29e-1**	**6.45e-4**	2.38e-1	1.21e-2
19a	3000	10000	2.20e-1	1.30e-3	**2.18e-1**	**2.20e-3**

C. Results Of Problem Instances Randomly Generated

To further evaluate the performance of FLAOS, we have tested it on larger problem instances, and compared it against the AOSEP. These problem instances have much larger numbers of requested items than the problem instances in [3], as well as more divers numbers of item types.

FLAOS and AOSEP are given the same number of cost function evaluations. The experimental results on the problem instances are summarized in Table IV. We can find that the mean best solutions of FLAOS are better than AOSEP's on fifteen problem instances except for problem instances 14, 18 and 19a.

Table V compares the results of two objectives (waste minimization and contiguity) for FLAOS and AOSEP on the randomly generated problem instances. As the total number

TABLE III. THE MEAN BEST RESULTS FOR PROBLEM INSTANCES 1-10 AND 1A-10A OVER 50 INDEPENDENT RUNS FOR CSPS WITH CONTIGUITY

inst	EP				FLAOS				AOSEP			
	Stocks used	Total wastage	Stocks w/ wastage	Max open	Stocks used	Total wastage	Stocks w/ wastage	Max open	Stocks used	Total wastage	Stocks w/ wastage	Max open
1	9.98	0	0	2.00	9.70	0	0	2.00	9.84	0	0	2.00
2	27.52	0	0	2.84	27.83	0	0	**2.65**	27.76	0	0	**2.65**
3	26.4	0	0	3.00	26.56	0	0	**2.3**	27.14	0	0	**2.29**
4	23.92	0	0	3.52	24.82	0	0	**2.98**	24.42	0.32	0.12	**2.92**
5	55.6	1	0.02	6.22	55.63	**0**	**0**	**5.81**	55.68	**0**	**0**	6.16
1a	9	3	2	2.00	9	3	2	2.00	9	3	2	2.00
2a	23	13	4.08	**2.30**	23	13	4	**2.46**	23	13	4.05	2.51
3a	15	0	0	4.26	15	0	0	**3.84**	15	0	0	**3.96**
4a	19	11	1.64	3.84	19	11	1.63	3.82	19	11	1.71	3.80
5a	53	**11450**	**23.56**	**6.88**	53.04	11625.51	24.42	**7.20**	53	11702.94	24.57	7.25
6	86.78	**4.84**	**2.46**	**8.32**	86.80	**3.36**	**1.78**	**8.86**	86.69	5.49	2.69	9.39
7	74.02	4.60	1.88	**11.12**	74.07	**4.09**	**1.13**	**11.09**	74.16	**4.00**	**1.14**	11.96
8	151.78	99.30	14.20	**16.18**	**151.01**	**25.88**	**2.87**	**17.57**	151.55	**67.47**	**7.51**	18.83
9	165.52	**7.40**	**3.90**	**19.46**	165.37	**6.80**	**3.90**	**19.95**	165.41	8.86	4.84	21.86
10	229.14	176.50	33.92	**24.16**	229.57	**121.42**	**25.39**	**28.57**	229.10	**134.41**	**28.63**	29.53
6a	80.76	**254.36**	33.70	**9.32**	80.94	270.13	**33.24**	**10.07**	80.80	**257.80**	33.90	11.06
7a	68.96	199.20	15.34	**11.88**	**68.12**	**152.23**	**9.09**	**12.74**	68.82	**182.82**	**10.71**	13.43
8a	148.08	**701.60**	76.68	**16.66**	**147.46**	**627**	**56.82**	**18.92**	148.33	732.00	**66.14**	19.69
9a	152.42	**432.40**	48.40	**20.10**	**151.93**	**395.84**	**33.96**	**23.17**	152.78	476.12	**39.98**	24.82
10a	220.28	643.60	70.70	**24.14**	**219.04**	**495**	**42.46**	**28.29**	219.79	**584.62**	**53.711**	29.25

TABLE V. THE MEAN BEST RESULTS FOR NEW PROBLEM INSTANCES 11-19 AND 11A-19A OVER 50 INDEPENDENT RUNS FOR CSPS WITH CONTIGUITY

inst	FLAOS					AOSEP				
	Stocks used	Total wastage	Stocks w/ wastage	Max open	Average open no.	Stocks used	Total wastage	Stocks w/ wastage	Max open	Average open no.
11	219.1	**49.5**	**21.3**	**9.5**	**6.9**	219	61	26.4	9.9	7.22
12	229.2	1	1	**35.2**	**26.47**	229.8	1	1	37.4	27.89
13	**289.93**	**2.87**	**1.68**	**61.56**	**44.73**	290.04	27	7.6	65.04	47.94
14	736.6	52840	571.6	10	**9.22**	**734**	**52820**	**565.2**	10	**8.87**
15	709.7	**29160**	**395.3**	40	36.68	709.9	36610	437.2	40	36.69
16	694.42	**26887.14**	**362.85**	79.71	**67.68**	**693.42**	34101.42	416.28	**79.14**	67.71
17	**1245.62**	**54687.5**	**540.12**	10	**9.83**	1247	69787.5	606.5	10	9.89
18	1248.71	79300	825.57	40	38.30	**1246.16**	**76566.66**	**815.33**	40	**38.14**
19	1289	**79430**	**855.5**	80	**72.22**	**1288.4**	96170	906.8	80	72.47
11a	207	**1647**	**114.7**	10	**8.48**	207.8	1819	129.5	10	8.64
12a	**214.8**	**1463**	**69.4**	**38.3**	**31.05**	215.81	1681.9	96.9	39.27	32.34
13a	**273.62**	**2245.37**	**144.25**	**70.87**	**53.75**	274.7	2476.5	159.5	71.4	54.08
14a	**698.4**	**385600**	**635.8**	10	9.71	701.2	410800	646.6	10	**9.69**
15a	**667.5**	**365990**	**569**	40	**37.01**	670.3	391190	584.6	40	37.19
16a	**653.6**	**365130**	**567.4**	**79.4**	**68.39**	656	386730	581	80	68.82
17a	**1171.37**	**724375**	**1068.37**	10	9.9	1174.7	754300	1076.8	10	**9.88**
18a	**1175.37**	**780975**	**1105.12**	40	**38.32**	1180.5	827100	1109.25	40	38.44
19a	**1213**	**766630**	1078	80	72.99	1214.33	778630	**1074**	80	**72.89**

of requested items become larger, the average number of maximum open items is usually equal to the types of items. The average number of open items of each stock is used to evaluate the objective of contiguity. For problem instance 14, AOSEP is especially better in terms of average open items of each stock (8.87 vs. 9.22). For problem instance 18, the total wastage of AOSEP is 76566.66, which is much less than the 79300 of FLAOS. For problem instance 19a, the total wastage of AOSEP is 778630, which is more than 766630 of FLAOS, but the average number of open items of AOSEP is 72.89, which is less than 72.99 of FLAOS. The number of stocks used of FLAOS is significant better than AOSEP on ten problem instances. As the results showed in Tables IV and V, FLAOS outperform AOSEP for the CSPs with contiguity. The fitness levels improve the adaptive operator selection and the performance of AOSEP.

D. Parameter Analysis

In FLAOS, 3PS and SRI mutations make the contributions to the performance. There are two other important parameters in FLAOS, the population size μ and the tournament size q. In both EP [3] and FLAOS, the $\mu = 75$ and the $q = 10$ are used. To investigate whether these two parameters have effect on performance of FLAOS or not, we test different values of these two parameters on problem instances 4, 5, 4a and 5a from [3]. These four problem instances were also used by the EP [3] to calibrate the parameter of number of 3PS mutation.

We assign 5, 10, 15, 20 to the tournament size q while

TABLE VI. THE COMPARE RESULTS OF DIFFERENT TOURNAMENT SIZE q

inst	Gen	Tournament size of FLAOS							
		5		10		15		20	
		Mean	Std. Dev	Mean	Std. Dev	Mean	Std. Dev	Mean	Std. Dev
4	2000	1.99e-2	5.45e-3	1.97e-2	5.54e-3	2.13e-2	6.77e-3	2.10e-2	6.92e-3
4a	2000	6.56e-2	7.42e-3	6.39e-2	6.93e-3	6.63e-2	7.12e-3	6.65e-2	7.07e-3
5	2000	6.77e-3	1.35e-3	6.91e-3	1.67e-3	6.57e-3	1.57e-3	6.53e-3	1.34e-3
5a	2000	1.28e-1	5.75e-3	1.29e-1	7.19e-3	1.29e-1	7.76e-3	1.27e-2	3.38e-3

TABLE VII. THE COMPARE RESULTS OF DIFFERENT POPULATION SIZE μ

inst	Gen	Population size of FLAOS									
		50		75		100		125		150	
		Mean	Std. Dev	Mean	Std. Dev	Mean	Std. Dev	Mean	Std. Dev	Mean	Std. Dev
4	2000	2.12e-2	6.83e-3	1.97e-2	5.54e-3	1.98e-2	5.09e-3	1.91e-2	1.35e-3	1.95e-2	4.73e-3
4	2000	6.76e-2	8.08e-3	6.39e-2	6.93e-3	6.56e-2	1.01e-2	6.41e-2	7.81e-3	6.46e-2	8.48e-3
5	2000	7.11e-3	2.28e-3	6.91e-3	1.67e-3	7.21e-3	1.69e-3	7.28e-3	1.57e-3	7.30e-3	1.45e-3
5a	2000	1.30e-1	7.42e-3	1.29e-1	7.19e-3	1.31e-1	7.87e-3	1.32e-1	9.77e-3	1.30e-1	6.38e-3

fixing the population size $\mu = 75$. We then assign 50, 75, 100, 150 to the population size while fixing $q = 10$. For the mean best results of each problem instance with different q or μ, we repeat the experiments for 50 independent runs.

Tables VI and VII presents the mean best fitness and the standard deviation of different tournament size q and population size μ. From the results, for every problem instance, the difference of the results of different tournament size q or population size μ is very small. We can conclude that the value of tournament size q or population size μ has little effect on the performance of FLAOS, i.e., FLOAS is a robust approach.

VI. CONCLUSIONS

In this study, FLAOS, an adaptive method to select proper operators for CSPs with contiguity is introduced. We propose the concept of *fitness level* which allows to adapt search operator usage probabilities for different individual types (in terms of fitness). Fitness levels are updated during the run. The adaptive method is analyzed by solving existing CSP instances from literature [3], [10] and 18 randomly generated CSP instances. The experimental study showed that FLAOS can achieve better performance than plain EP and than adaptation without fitness levels.

In FLAOS, a method to generate new fitness level is introduced. When new individuals are generated, their fitness values may be better than the lower value of the current best fitness levels. When the number of these new better individuals exceeds a certain threshold, we create a new fitness level whose interval is from the fitness value of the best individual to the lower fitness value of previous best fitness level. This method can ensure that the adaptation of operator selection probabilities can appropriately cover the whole spectrum of fitness values in whole population.

There are some improvements which can be applied to FLAOS in the future. For example, we can use a non-generational replacement strategy that each individual has a life time, which decides the survival of individual [27]. We can further want to investigate the application of FLAOS [6] in scenarios where more than two search operators are involved.

REFERENCES

[1] S. P and Paternoster, "One-dimensional cutting stock decision packing problems," *Journal of the Operational Research*, vol. 43, pp. 691–706, 1992.

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

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

[4] D. H and F. U, *Cutting and packing in production and distribution: a typology and bibliography*. Heidelberg: Physica-Verlag, 1992.

[5] G. Wäscher, H. Haubner, and H. Schumann, "An improved typology of cutting and packing problems," *European journal of operational research*, vol. 183, pp. 1109–1130, 2007.

[6] R. Chiong and O. K. Beng, "A comparion between genetic algorithms and evalutionary programming based on cutting stock problem," *Engineering Letters*, vol. 14, pp. 72–77, 2007.

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

[8] Á. Fialho, L. D. Costa, M. Schoenauer, and M. Sebag, *Analyzing Bandit-based Adaptive Operator Selection Mechanisms*. Springer Netherlands, 2010.

[9] Tuson, A. Ross, and P., "Adapting operator settings in genetic algorithms," *Evolutionary Computation*, vol. 6(2), pp. 161–184, 1998.

[10] R. Hinterding and L. Khan, *Genetic Algorithms for Cutting Stock Problems: with and without Contiguity*. Springer Berlin Heidelberg, 1995.

[11] P.C.Gilmore and R.E.Gomory, "A linear programming approach to the cutting-stock problem," *Operations Research*, vol. 9, pp. 849–859, 1961.

[12] P.Gilmore and R.Gomory, "A linear programming approach to the cutting-stock problem - part ii," *Operations Research*, vol. 11, pp. 863–888, 1961.

[13] B. J. Yuen, "Heuristics for sequencing cutting patterns," *European Journal of Operational Research*, vol. 55, pp. 183–190, 1991.

[14] K. C. Poldi and M. N. Arenales, "Heuristics for the one-dimensional cutting stock problem with limited multiple stock lengths," *Computers and Operations Research*, vol. 36, pp. 2074–2081, 2009.

[15] M. Gradisar and P. Trkman, "A combined approach to the solution to the general one-dimensional cutting stock problem," *Computers Operations Research*, vol. 32, pp. 1793–1807, 2005.

[16] Y. Cui and Y. Yang, "A heuristic for the one-dimensional cutting stock problem with usable leftover," *European Journal of Operational Research*, vol. 196, pp. 245–250, 2008.

[17] A. C. Cherri, M. N. Arenales, and H. H. Yanasse, "The one-dimensional cutting stock problem with usable leftover - a heuristic approach," *European Journal of Operational Research*, vol. 196, pp. 897–908, 2009.

[18] A. C. Cherri and M. N. Arenales, "The usable leftover one-dimensional

cutting stock problem–a priority-in-use heuristic," *International Transactions in Oprational Research*, vol. 20, pp. 189–199, 2013.

[19] H. Reinertsen and T. W. Vossen, "The one-dimensional cutting stock problem with due dates," *European Journal of Operational Research*, vol. 201, pp. 701–711, 2010.

[20] H. H. Yanasse and M. J. P. Lamosa, "an integrated cutting stock and sequencing problem," *European Journal of Operational Research*, vol. 183, pp. 1353–1370, 2007.

[21] C. T. Ragsdale and C. W.Zobel, "The ordered cutting stock problem," *Decision Sciences*, vol. 35, pp. 83–100, 2004.

[22] G. Eiben and M. C.Schut, *New Ways to Calibrate Evolutionary Algorithms*. Springer Berlin Heidelberg, 2007.

[23] Á. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. Smith, *Parameter Control in Evolutionary Algorithm*. Springer Berlin heidelberg, 2007.

[24] B. M. Ginley, J. Maher, C. O'Riordan, and F. Morgan, "Maintaining healthy population diversity using adaptive crossover, mutation and selection," *IEEE Transations on Evolutionary Computation*, vol. 15, pp. 692–714, 2011.

[25] F. D., *Evolutionary computation: toward a new philosophy of machine intelligence*. New York: IEEE Press, 1995.

[26] P.Vaja, A. Eiben, and W. Hordijk, *Parameter Control Methods for Selection Operators in Genetic Algorithm*. PPSN X, 2008.

[27] J. Arabas, Z. Michalewicz, and J. Mulawka, "Gavaps - a genetic algorithm with varying population size," in *Evolutionary Computation*, vol. 1, 1994, pp. 73–78.