Yugoslav Journal of Operations Research
31 (2021), Number 3, 285–298
DOI: https://doi.org/10.2298/YJOR200117009K

A HYBRID VNS MATHEURISTIC FOR A BIN PACKING PROBLEM WITH A COLOR CONSTRAINT

Yury KOCHETOV
Sobolev Institute of Mathematics, 4 Koptyug ave., 630090 Novosibirsk Russia
jkochet@math.nsc.ru

Arteam KONDAKOV
Novosibirsk State University, 1 Pirogova st., 630090 Novosibirsk Russia
tyxtyxyc@gmail.com

Received: January 2020 / Accepted: January 2021

Abstract: We study a new variant of the bin packing problem with a color constraint. Given a finite set of items, each item has a set of colors. Each bin has a color capacity, the total number of colors for a bin is the unification of colors for its items and cannot exceed the bin capacity. We need to pack all items into the minimal number of bins. For this NP-hard problem, we present approximability results and design a hybrid matheuristic based on the column generation technique. A hybrid VNS heuristic is applied to the pricing problem. The column generation method provides a lower bound and a core subset of the most promising bin patterns. Fast heuristics and exact solution for this core produce upper bounds. Computational experiments for test instances with number of items up to 500 illustrate the efficiency of the approach.

Keywords: VNS, Matheuristic, Local Search, Large Neighborhood, Column Generation.
MSC: 90B85, 90C26.

1. INTRODUCTION

In the classical bin packing problem, a set of weighted items must be packed into the minimal number of identical bins such that the sum of weights of items

Preliminary version of the paper was presented at the 4th International Conference on Variable Neighborhood Search [20].

in each bin does not exceed capacity of the bin. In this paper we study a new variant of the bin packing problem. Each item does not have the weight but has a set of colors. The bin capacity limits the total number of colors for its items. The goal is to pack all items into the minimal number of identical bins such that the total number of colors of items in each bin does not exceed the bin capacity. It is NP-hard problem in the strong sense and can be reformulated as a biclique vertex covering problem for bipartite graphs [14].

The bin packing problem with color constraints (BPC) presents a recent line of research in combinatorial optimization. One of its applications is beverage package printing [22]. In [25] the bin packing problem with classes of items (colors) is used to model video-on-demand applications. A biclique covering (biclustering), which is a straight reformulation of BPC, is used to model protein-protein interaction [11].

There are some versions of the problem with online and offline settings [6, 23]. In the colored bin packing [10], each bin has a maximum color capacity, i.e., a limit on the number of items of a particular color. This version originates from the production planning of a steel plant. In a more generalized version of the color bin packing problem [18, 21], the constraints among items are described by a conflict graph. In the black and white bin packing problem with alternation constraints [5], two items with the same color can not be packed adjacently to each other.

In this paper we consider a new version of the colored bin packing problem, which is a special case of the co-printing problem [22]. We assume that each item has zero weight and a set of colors. We apply the column generation method for the model with exponential number of variables. It produces a lower bound. The VNS matheuristic with large neighborhoods is designed for solving the pricing problem and accelerating the method. The column generation method provides a core subset of the most promising bin patterns. This core is used to get upper bounds. Similar idea was suggested by P. Avella et al. [4] for solving large scale p-median problem by Lagrangian relaxations. We apply three heuristics and an exact method (GUROBI) to this core. It is interesting to note that the exact method is very efficient in this case and surpasses the heuristics. We need a lot of running time for column generation method and a small amound of time for getting upper bound by exact method. In our computational experiments, we observe that the exact method provides good results even for a subcore when we terminate the column generation at an intermediate step. We illustrate this useful idea on computational experiments for the large scale instances with number of items up to 500. We study the efficiency of our hybrid approach on randomly generated test instances. For that case, this hybrid method has found optimal solutions.

The paper is organized as follows. In Section 2 we introduce notations and present the mathematical model. In Section 3 we present approximability results. We show that the well-known set covering problem can be reduced to BPC problem. Thus, it is hard for approximation in polynomial time. In Section 4 we discuss the column generation method and describe the pricing problem. In Section 5 we

define three types of large neighborhoods and present a pseudocode of the hybrid VNS matheuristic for the pricing problem. In Section 6 we design heuristics for the BPC problem. Finally, in Section 7 we discuss experimental results for randomly generated test instances and instances with regular structure. Summary and conclusions are in Section 8.

2. MATHEMATICAL MODEL

Let us introduce the following notations:
$I = \{1, \ldots, n\}$ is the set of items;
$J = \{1, \ldots, m\}$ is the set of colors;
$K_i \subset J$ is the set of colors for item i;
b is the upper bound for number of different colors in each bin;
$p = (p_1, \ldots, p_i, \ldots, p_n)$ is a bin pattern, or bin for shot, where $p_i \in \{0, 1\}$ denotes whether item i is in the bin or not. We consider only the feasible patterns that satisfy the bin capacity constraint: $|\cup (K_i : p_i = 1)| \leq b$.

Now we can write the BPC problem as follows:

$$\min \Big\{ \sum_{p \in P} y_p \ : \ \sum_{p \in P} p_i y_p \geq 1, i \in I, \ y_p \in \{0, 1\} \Big\}. \tag{1}$$

In this linear integer programming formulation, we have a lot of variables and a few constraints. The large scale formulations allow us to exclude symmetries, which are usually presented in the bin packing compact representations. Moreover, large scale formulations as a rule have small integrality gap and the linear programming relaxation can be solved exactly by column generation technique, which is described in Section 4.

If we assume that all colors are different for any pair of items, then we get the classical bin packing problem. Thus, the BPC problem is NP-hard in the strong sense. We have polynomial time heuristics with constant performance guarantee for this problem. Unfortunately, the BPC problem is harder for approximation in polynomial time. As we will see in Section 3, the minimum hitting set problem can be reduced to the BPC problem and this reduction preserves approximability. Hence, the BPC problem is not approximable within $(1 - \varepsilon) \ln b$ for arbitrary positive ε unless $P = NP$. In particular, the BPC problem does not have polynomial time algorithm with constant performance guarantee for arbitrary large constant.

Another interesting formulation of the BPC problem can be done in terms of bipartite clique covering. Let $G(V_1, V_2, E)$ be the bipartite graph with sets of vertices V_1, V_2 and set of edges $E \subset V_1 \times V_2$. A biclique K_{st} in G is a complete bipartite subgraph with s vertices from V_1 and t vertices from V_2. We wish to cover graph G by a minimal number of bicliques for large $t, t \geq D$ for a given threshold D. It is easy to see that the BPC problem is equivalent to this biclique covering problem. We put $V_1 = I$, $V_2 = J$ and an edge (ij) belongs to E if and only if $j \notin K_i$. Each feasible bin in the BPC problem is a biclique in the covering problem for $D = |J| - b$ and vice versa.

3. APPROXIMABILITY RESULTS

Let us consider an important special case of the BPC problem when $b = m-1$. We will see that the case is equivalent to the well-known Minimum Hitting Set problem (MHS). Remember that in the MHS problem we have a finite set S and a collection C of its subsets $S_i, i = 1, \ldots, t$, where $t = |C|$. We need to find a hitting set with minimal cardinality, i.e., a minimal subset $S' \subseteq S$ such that S' contains at least one element from each subset in C.

Theorem 1. *The MHS problem is equivalent to the BPC problem with* $b = m-1$.

Proof. Assume that we have an instance for the MHS problem. We create an instance for the BPC problem by the following rule:

$$J = S,\ I = C,\ K_i = S \setminus S_i, i \in C,\ b = |S| - 1.$$

Now we consider a hitting set S' and a corresponding cover of the collection C by subcollections $C_j,\ j \in S'$. We assume that subset S_i belongs to C_j if S_i contains element j. According to our rule, each subset S_i corresponds to an item i for the BPC instance and $K_i = S \setminus S_i$. Hence, the union of colors for all items from C_j does not contain color j. As a result, we can put all these items in a bin. Thus, we have a solution for the BPC problem with the same value of the objective function.

Assume that we have an instance for the BPC problem. We create an instance for the MHS problem by a similar rule:

$$S = J,\ C = I,\ S_i = J \setminus K_i, i \in I.$$

In a feasible solution for the BPC problem, each bin has as most $b = m-1$ colors. It means that at least one color is not contained in it and we can collect all such colors as a hitting set. As a result, we have got a solution for the MHS problem with the same or better value of the objective function. □

Note that the above reductions are linear and preserve approximability, and in fact realize so called L-reductions [2]. In other words, all approximation algorithms for the MHS problem will carry over to the BPC problem with $b = m-1$. Moreover, nonapproximability results will carry over to the general case of BPC problem as well.

Corollary 2. *For the BPC problem with* $b = m-1$*, the greedy algorithm produces an* $H(b)$*–approximate solution, where* $H(b) = \sum_{i=1}^{b} 1/i$.

Proof. The MHS problem is equivalent to the Minimum Set Cover problem [3]. Hence, the greedy algorithm of Chvatal [7] has the desired property. □

Corollary 3. *The BPC problem is not approximable within* $(1-\varepsilon)\ln b$ *for any* $\varepsilon > 0$ *unless* $P = NP$.

Proof. This negative result is followed from similar property for the Minimum Set Cover problem [12]. □

4. COLUMN GENERATION PROCEDURE

According to the well-known column generation approach we restrict ourselves to a small subset $P' \subset P$, initially generated by some heuristic. Then at each iteration, a pricing problem is solved, and new columns with negative reduced cost are generated. We add all such columns to the restricted master problem. Now we consider the dual linear programming problem in relation to the master problem:

$$\max\Big\{\sum_{i\in I} w_i \ : \ \sum_{i\in I} p_i w_i \le 1, p \in P', \quad w_i \ge 0\Big\},$$

where the dual variable w_i can be considered as a price for item i. To enlarge the subset P' or terminate the method, we should solve the following pricing problem with optimal values w_i^* of the dual variables.

Let us introduce additional variables:
$x_i = 1$ if item i is placed in a bin and $x_i = 0$ otherwise,
$z_j = 1$ if a bin contains item with color j and $z_j = 0$ otherwise.

Then, the pricing problem can be stated as follows:

$$\min(1 - \sum_{i\in I} w_i^* x_i)$$

$$\text{s.t.} \sum_{j\in J} z_j \le b;$$

$$x_i \le z_j, \quad j \in K_i, i \in I;$$

$$x_i, z_j \in \{0,1\}, \quad i \in I, j \in J.$$

The objective function minimizes the reduced cost. The first constraint controls the total number of colors for a bin. The second constraint shows the relations between items and colors. It is easy to see that a knapsack problem can be reduced to the pricing problem, and as a result, the pricing problem is NP-hard. We have to solve a lot of the pricing problems with different prices. Thus, we design a VNS matheuristic to accelerate the method. A block diagram for the column generation procedure is presented in Figure 1.

To reduce the number of iterations of the column generation method, we introduce *deep dual-optimal inequalities* [15] to the restricted master problem. It is a set of restrictions, which, if added to dual problem, does not cut off at least one optimal solution. The following proposition establishes a foundation for our inequalities:

Proposition 4. *Let I_0, I' be two disjoint sets of items such that $\cup_{i\in I_0} K_i \supseteq K_{i'}$ for all $i' \in I'$. Then there exist a dual optimal solution w^* for the restricted master problem such that $\sum_{i\in I_0} w_i^* \ge \sum_{i'\in I'} w_{i'}^*$.*

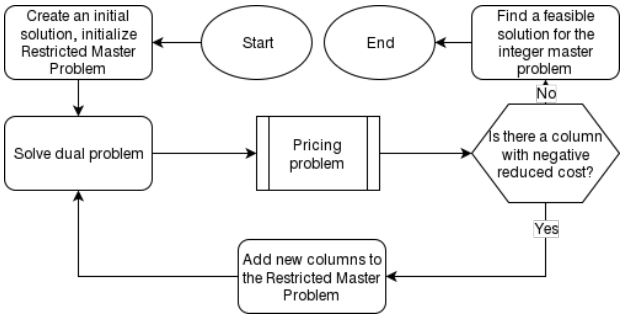

Figure 1: Block diagram of column generation algorithm

Proof. It is easy to verify that the problem (1) has the (s,t) exchange property [15] for $s_i = 1$ if $i \in I_0$, $t_i = 1$ if $i \in I'$ and 0 otherwise. Indeed, in that case, the set of items I_0 can be replaced with all items from I' without breaking the capacity constraint. Therefore, using Proposition 2 from [15], we conclude that $\sum_{i \in I_0} w_i^* \geq \sum_{i' \in I'} w_{i'}^*$ is the dual-optimal inequality. □

At each iteration, we identify a set of items I_0 for which the following inequality

$$\sum_{i \in I_0} w_i^* < \sum_{i \in \{i | K_i \subseteq \cup_{i' \in I_0} K_{i'}\} \setminus I_0} w_i^* \tag{2}$$

holds and add it to the master problem. We will demonstrate the efficiency of those inequalities in Section 7.

5. HYBRID VNS MATHEURISTIC

Variable Neighborhood Search is a well-known metaheuristic for combinatorial and global optimization based upon systematic change of neighborhood within local search [17]. Its development has been successful in many real-world applications, including Stackelberg games ([1], [9]).

Below we apply this approach to the pricing problem. For a feasible solution (x_i^0, z_j^0), we put $K^0 = \cup\{K_i \mid x_i^0 = 1\}$ and define three types of neighborhoods.

- k-ItemAdd neighborhood. We collect all items with at most k additional colors to K^0 and select one of them, say i'. Then we create a subset $K' = K^0 \cup K_{i'}$ and a subset of items $I' = \{i \in I \mid K_i \subseteq K'\}$. The neighboring solution is defined as the optimal solution to the pricing problem with the following restriction: $x_i = 0, i \notin I'$. The size of the neighborhood is $O(n)$. Note that optimal solution in this definition is equivalent to the best solution in large neighborhood defined by this restriction.

- k-ColorDel neighborhood. We select a subset of colors $K' \subseteq K^0, |K'| \leq k$. The neighboring solution is defined as the optimal solution to the pricing

problem with the following restriction: $z_j = 1, j \in K^0 \setminus K'$. The size of the neighborhood is exponential.

- k-ColorSwap neighborhood. We remove k colors from the set K^0 and include other k colors resulting in a set K'. The neighboring solution is defined as all items with colors in K'. The size of the neighborhood is exponential, but each neighboring solution can be evaluated in polynomial time.

To get neighboring solutions for the k-ItemAdd and k-ColorDel neighborhoods, we need to solve the pricing problems. For small k they have small dimension and can be solved easily by Gurobi software [16]. Resolution of pricing problem in full dimension is time consuming.

Note that, after performing any jump in k-ItemAdd neighborhood, we found the best solution among all solutions, where the set of colors is a subset of $K' = K^0 \cup K_{i'}$. Hence, we can exclude all these solutions from consideration. So, we collect the subsets K' during the search by the k-ItemAdd neighborhood and include the cuts

$$\sum_{j \in K'} z_j \leq b - 1 \tag{3}$$

into the pricing problem to remove them. For the k-ColorDel neighborhood the similar property holds, and we use the cuts

$$\sum_{j \in J \setminus (K^0 \setminus K')} z_j \leq k - 1. \tag{4}$$

The cuts are accumulated in the TabuList with fixed length. The pseudocode of the hybrid VNS and Tabu Search matheuristic is presented below.

Hybrid VNS matheuristic

1. Select the set of neighborhood structures for $k = 1, \ldots, k_{max}$, $s = 1, \ldots, s_{max}$; find an initial solution (x, z); put TabuList := $\emptyset$; choose a stopping condition T_{max};

2. Set $T := 1$

3. Repeat the following sequence until the stopping condition $T = T_{max}$ is met:

(1) Set $k := 1; s := 1$;

(2) If T is even:

(a) Repeat the following steps until $k = k_{max}$:

(a.1) Select a solution (x', z') at random by the k-ItemAdd neighborhood

(a.2) Update Tabu List: add restriction of type (3), induced by selected solution

(a.3) If solution (x', z') is better than incumbent solution (x, z) then move $(x, z) := (x', z')$; $T := 1$.

(3) If T is odd:

(a) Repeat the following steps until $s = s_{max}$:

(a.1) Select a solution (x', z') at random by the s-ColorDel neighborhood

(a.2) Update Tabu List: add restriction of type (4), induced by selected solution

(a.3) If solution (x', z') is better than incumbent solution (x, z) then move $(x, z) := (x', z')$; $T := 1$.

(4) $T := T + 1$;

(5) If we have no improvement for $\frac{T_{max}}{2}$ iterations then move by the k-ColorSwap neighborhood at random direction;

4. Return the best found solution (x, z).

The block diagram for this algorithm is presented in Figure 2.

It is usually harder to get a solution from k-ColorDel neighborhood, so the parameter s_{max} which controls its size is usually smaller than corresponding parameter k_{max} for k-ItemAdd neighborhood.

To reduce computation time, we can reduce the size of the problem. The items with large color sets K_i are the most inconvenient for the heuristic. So,

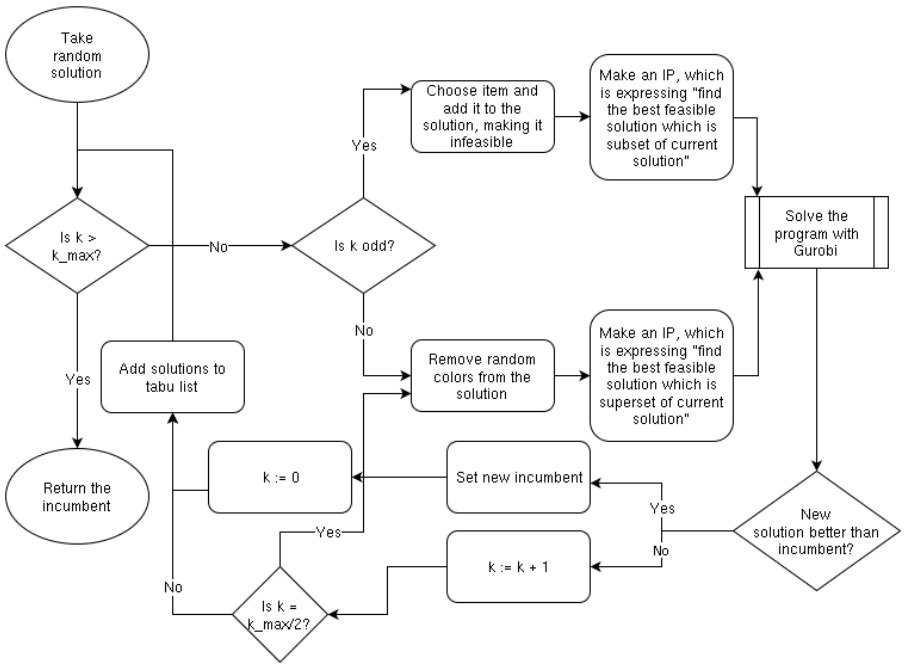

Figure 2: Block diagram of the hybrid matheuristic

we select all large items, $|K_i| \geq b - r$, and solve the pricing problem for each of them with additional constraint $x_i = 1$. Since those items have a lot of colors, resulting problem is, in fact, pretty small. Then we apply the hybrid matheuristic for the remaining items. As we noted before, we include all founded solutions with negative reduced cost into the restricted master problem. In such a way, we decrease the number of iterations but increase the total number of columns. If the matheuristic cannot find a solution with negative reduced cost, then we solve the pricing problem to optimality by commercial solver Gurobi.

6. HEURISTICS FOR THE BPC

We apply four heuristics to find optimal or near optimal solutions:

- The FFD heuristic. It is an adaptation of the well known FFD algorithm for the classical bin packing problem. The items are sorted by the nonincreasing of the cardinality of their color sets. Then we put each item in the first bin it fits in. If it does not fit in any bin, we open a new bin.

- The FillBin heuristic. This heuristic has been inspired by the $FillBin$ heuristic from [24]. We put $\overline{w}_i = |K_i|^2$ and use these artificial prices in the pricing problem. The hybrid VNS matheuristic is applied to fill the first bin. The packed items are removed and this matheuristic is applied again to fill the

second bin. After that, we fill the third bin, fourth bin, and so on until we pack all items.

- The LP heuristic. We use the following idea [22] at the last iterations of the column generation procedure. We take the round down LP solution to the restricted master problem as a partial solution and apply the FillBin heuristic to the remaining items. The best found solution is returned as the result of the LP heuristic.

- The IP heuristic After the column generation procedure terminates, we seek for optimal integer solution of the restricted master problem using Gurobi optimizer. A similar approach has been introduced in core heuristic in [4]. Instead of reducing full problem to its core, column generation allows us to pull the core without considering full problem.

LP heuristic is time consuming but produces strong solutions. The IP heuristic is faster than LP heuristic, and in our experiments it nearly always produces solutions that match lower bound. Moreover, we can apply LP and IP heuristics if the column generation is terminated in an intermediate iteration. Sure, in such a case we have not a lower bound for the global optimum, but IP heuristic produce solutions, equal by their cost to current rounded-up restricted master problem solution.

The FFD and FillBin heuristics are fast but produce weak solutions. We use these heuristics at the initialization step of the VNS and column generation procedure.

7. COMPUTATIONAL EXPERIMENTS

We conduct our computational experiments for the randomly generated test instances in 10 sets of parameters n, m and b. For each set, we generate 10 instances. Thus, the total amount of tests is equal to 100. The color set K_i for each item is generated by the following procedure. We uniformly choose an integer l from 1 to b and uniformly choose a 0-1 vector with exactly l ones.

We use $k_{max} = 16$, $s_{max} = 8$, $T_{max} = 10$, $r = 0.2b$ as values of parameters in our experiments. The TabuList has the constant length equals 10.

Table 1 presents the results of our experiments for different variants of heuristic. Column It is averaged amount of column generation iterations among 10 instances, column C is average amount of generated columns, and column T is average time (in seconds). Columns $v1$ show the performance of heuristic with all features that have been described earlier; $v2$ show a heuristic without stabilization inequalities (2); $v3$ show a heuristic in which the hybrid VNS matheuristic for pricing problem is performed without prior big items consideration. Every variant uses an IP heuristic at the final stage for the whole problem. For all instances, lower bounds coincide with upper bounds and we have got the optimal solutions. The Gurobi solver spends a lot of running time at the last iteration to prove that there is no column with negative reduced cost.

			$v1$			$v2$			$v3$		
n	m	b	It	C	T	It	C	T	It	C	T
150	55	30	87	261	2256	110	313	2601	89	270	2893
175	55	30	122	294	2691	143	339	3055	124	302	2914
200	55	30	98	316	2772	132	342	3258	105	312	3151
225	55	30	119	322	3024	148	361	3406	120	331	3383
250	55	30	141	347	4017	170	405	4249	141	341	4373
150	90	40	101	277	2318	128	320	2913	105	282	2766
175	90	40	125	305	2652	151	334	3122	124	317	3078
200	90	40	136	329	2910	154	357	3289	136	335	3166
225	90	40	156	362	3573	183	398	4170	157	367	3884
250	90	40	202	481	5837	257	530	6714	210	486	6526

Table 1: Computational results

We note that the performance of the method strongly depends on the integrality gap and the bin capacity. For instances without the gap, our approach shows good performance in running time. It seems that the uniformly generated instances are quite *easy* for the method. Many of them have not the integrality gap in the large scale formulation. If the integrality gap is positive, we need a lot of efforts to solve the linear programming relaxation. Nevertheless, we can apply the IP heuristic in such a case as well even for large bin capacity and before the termination of column generation.

Note that the previous studies on the BPC [22] deal with small bin capacity only, $b \leq 10$. An addaptation of the FFD heuristic, the $FillBin$ heuristic and the LP heuristic has been studied. In [24] an original $FillBin$ heuristic have been used for classical bin packing problem. The classical FFD algorithm for bin packing problem has been widely studied [8].

Table 2 compares performance of four heuristics for BPC, described in Section 6. We use the same instances as for Table 1. Columns Acc show the average percentage deviation from the column generation lower bound, and columns T show the average running time for each heuristic (in seconds). Note that for IP and LP heuristics the column generation procedure must be performed, thus, we exclude that time from the results.

n	m	b	FFD		$FillBin$		LP		IP	
			Acc	T	Acc	T	Acc	T	Acc	T
150	90	40	45.1%	0.22	31.25%	124	1.25%	62	0%	2.31
175	90	40	47.32%	0.29	30.82%	157	2.93%	81	0%	2.85
200	90	40	47.63%	0.31	35.14%	195	4.24%	101	0%	2.72
225	90	40	50.88%	0.34	36.75%	219	5.31%	118	0%	3.2
250	90	40	49.94%	0.36	38.53%	262	5.85%	141	0%	3.58

Table 2: Comparison of the heuristics

A feasible solution for BPC can be obtained at each iteration of the column generation procedure. We use this approach to find a heuristic solution for large scale instances. Accuracy of those solutions can be estimated with intermediate lower bound [13]: given an optimal cost Z^*_{RMP} for the restricted master problem, an optimal solution w for the dual-master problem, an optimal solution x^* for the pricing problem, the following inequality hold:

$$\frac{Z^*_{RMP}}{w^\top x^*} \leq Z^*_{MP} \tag{5}$$

where Z^*_{MP} is an optimal cost for the full master problem.

We terminate our heuristic when the hybrid matheuristic for the pricing problem can't find a solution with negative reduced cost. Then we obtain the solution for BPC by applying the IP heuristic. After that, we solve the pricing problem with Gurobi to get the lower bound (5). We observe that this intermediate lower bound is pretty rough: if we run column generation to the end, a solution improves just a little, while lower bound improves significantly. However, it is hard to get the optimal solution of the pricing problem.

n	m	b	It	C	T_{CG}	T_{IP}	Acc_{LB}	Acc_{CG}
150	90	40	28	232	416	2.12	8.03%	0%
175	90	40	41	257	621	2.61	8.1%	0.25%
200	90	40	65	281	1683	2.58	9.25%	1.33%
225	90	40	86	314	2258	3.13	11.38%	1.2%
250	90	40	93	384	3104	3.6	12.6%	1.56%
300	90	40	97	426	4841	4.21	15.33%	-
350	90	40	95	485	6103	4.88	20.6%	-
400	90	40	98	515	7057	5.37	21.14%	-
450	90	40	101	648	8508	6.16	23.78%	-
500	90	40	103	810	10736	7.71	35.53%	-

Table 3: Large instance computational results

Table 3 presents the results for that truncated heuristic. Columns n, m, b, It, C have the same meaning as for Table 1, column Acc_{LB} shows the average percentage deviation of heuristic solutions from the intermediate lower bound, column Acc_{CG} shows the average percentage deviation from the optimum. Columns T_{CG} and T_{IP} show the running time of column generation procedure and IP heuristic respectively. Again, the column generation procedure is the most time consuming. Optimal integer solution for the restricted master problem can be founded easily.

8. CONCLUSIONS

In this paper we studied a new variant of the bin packing problem with a color constraint for bin capacity. We have shown that this problem is hard to approximate in polynomial time. We designed a column generation based matheuristic.

The hybrid VNS approach is used for the pricing problem to accelerate the method. For further research, it would be interesting to find difficult test instances where the proposed method produces solutions with high deviation from the optimum. For the classical bin packing problem, we have very interesting open question about the quality of the lower bound produced by the column generation approach. As a rule, the gap between optimal solution and this lower bound is 0 or 1. We still have no instance with the gap greater than 1. Is it possible to find such difficult instances for this new bin packing problem? It is an open question for future research.

We plan to investigate the performance of the method for the regular instances, where each item has the same number of colors and each color is included into the same number of items. Similar ideas are used for creating difficult test instances for the facility location problem [19]. Randomly generated test instances, where each facility has the same number of potential clients and each client can be serviced by the same number of facilities, are the most difficult from the computational point of view. Nevertheless, another regular class of test instances based on the finite projective planes are polynomially solvable. Another line of research is to construct an exact method for the BPC problem.

Acknowledgement: This research has been supported by RFBR grant (N 18-07-00599).

REFERENCES

[1] Alekseeva, E., Kochetov, Yu., and Plyasunov, A., "An exact method for the discrete $(r|p)$-centroid problem", *Journal of Global Optimization*, 63 (3) (2015) 445–460.

[2] Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M., *Complexity and approximation: Combinatorial optimization problems and their approximability properties*, Springer Science & Business Media, 2012.

[3] Ausiello, G., D'Atri, A., and Protasi, M., "Structure preserving reductions among convex optimization problems", *Journal of Computer and System Sciences*, 21 (1) (1980) 136–153.

[4] Avella, P., Boccia, M., Salerno, S., and Vasilyev, I., "An aggregation heuristic for large scale p-median problem", *Computers & Operations Research*, 39 (7) (2012) 1625–1632.

[5] Balogh, J., Békési, J., Dosa, G., Kellerer, H., and Tuza, Z., Black and white bin packing, *Approximation and Online Algorithms*, Springer, 2012, 131–144.

[6] Böhm, M., Sgall, J., and Veselỳ, P., Online colored bin packing, In: Bampis E. Svensson O. (eds) *Approximation and Online Algorithms*, WAOA 2014, Lecture Notes in computer Scince, vol 8952, Springer, Cham., 35–46.

[7] Chvatal, V., "A greedy heuristic for the set-covering problem", *Mathematics of Operations Research*, 4 (3) (1979) 233–235.

[8] Coffman Jr, E.D., Garey, M.R., and Johnson D.S., Approximation algorithms for bin packing: a survey, *Approximation algorithms for NP-hard problems*, pages 46–93. PWS Publishing Co., Boston, MA, Unated States, 1996.

[9] Kochetov, Yu., Davydov, I., and Carrizosa, E., "A local search heuristic for the $(r|p)$-centroid problem in the plane", *Computers & Operations Research*, 52 (B) (2014) 334–340.

[10] Dawande, M., Kalagnanam, J., and Sethuraman J., "Variable sized bin packing with color constraints", *Electronic Notes in Discrete Mathematics*, 7 (2001) 154–157.

[11] Ding, C., Zhang, Ya., Li, T., and Holbrook, S. R., Biclustering protein complex interactions with a biclique finding algorithm, *Sixth International Conference on Data Mining (ICDM'06)*, Hong Kong, China, IEEE, 2006, 178–187.

[12] Dinur, I., Steurer, D., Analytical approach to parallel repetition, *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, ACM, New York, NY, United States, May 2014, 624–633,
[13] Farley, A.A., "A note on bounding a class of linear programming problems, including cutting stock problems", *Operations Research*, 38 (5) (1990) 922–923.
[14] Fleischner, H., Mujuni, E., Paulusma, D., and Szeider, S., "Covering graphs with few complete bipartite subgraphs", *Theoretical Computer Science*, 410 (21) (2009) 2045–2053.
[15] Gschwind, T., and Irnich, S., "Dual inequalities for stabilized column generation revisited", *INFORMS Journal on Computing*, 28 (1) (2016) 175–194.
[16] Inc. Gurobi Optimization, Gurobi optimizer reference manual, 2015.
[17] Hansen, P., and Mladenović, N., "Variable neighborhood search: Principles and applications", *European Journal of Operational Research*, 130 (3) (2001) 449–467.
[18] Jansen, K., "An approximation scheme for bin packing with conflicts", *Journal of Combinatorial Optimization*, 3 (4) (1999) 363–377.
[19] Kochetov, Yu.A., Facility location: discrete models and local search methods. In Chvatal V., editor, Combinatorial Optimization. Methods and Applications, IOS Press, Amsterdam, Netherlands, 2011, 97–134.
[20] Kochetov. Yu.A., and Kondakov, A.A., "VNS matheuristic for a bin packing problem with a color constraint", *Electronic Notes On Discrete Mathematics*, 58 (2017) 39–46.
[21] Muritiba, A.E.F., Iori, M., Malaguti, E., and Toth, P., "Algorithms for the bin packing problem with conflicts", *Informs Journal on Computing*, 22 (3) (2010) 401–415.
[22] Peeters, M., and Degraeve, Z., "The co-printing problem: A packing problem with a color constraint", *Operations Research*, 52 (4) (2004) 623–638.
[23] Shachnai, H., and Tamir, T., "Polynomial time approximation schemes for class-constrained packing problems", *Journal of Scheduling*, 4 (6) (2001) 313–338.
[24] Vanderbeck, F., "Computational study of a column generation algorithm for bin packing and cutting stock problems", *Mathematical Programming*, 86 (3) (1999) 565–594.
[25] Xavier, E.C., and Miyazawa, F.K., "The class constrained bin packing problem with applications to video-on-demand", *Theoretical Computer Science*, 393 (1) (2008) 240–259.