2019 China-Qatar International Workshop on Artificial Intelligence and Applications to Intelligent Manufacturing (AIAIM)
January 1-4, 2019, Doha, Qatar

Comparison of Different Grasp Algorithms for The Heterogeneous Vector Bin Packing Problem

Đorđe Stakić
Faculty of Economics
University of Belgrade
Kamenička 6, 11000 Belgrade, Serbia
Email: djordjes@ekof.bg.ac.rs

Ana Anokić
Independent researcher
Branka Pešića 27
11000 Belgrade, Serbia
Email: anokicana@gmail.com

Raka Jovanovic
Qatar Environment and
Energy Research Institute (QEERI)
Hamad bin Khalifa University
PO Box 5825, Doha, Qatar
Email: rjovanovic@hbku.edu.qa

***Abstract*—In this paper, we address the practical problem of packing multiple items into containers for further transport. Dense packing of containers can significantly decrease supply chain costs, since transport fees are related to the the number of used containers and not the content. This practical problem is generally modeled using the vector bin packing problem (VBPP) and its variations. In the recent years, the heterogeneous VBPP with two sets of constraints has proven to be a good representation of container packing related problems. In this work we extend this model to a more realistic setting, by allowing multiple containers of the same type. To solve this problem, an integer program is designed. To be able to find feasible solutions for large scale problem instances, a greedy constructive algorithm is developed. With the intention of improving the solutions generated in this way, a local search is developed and used to extend the greedy algorithm to the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic. To evaluate the potential of using GRASP on this problem, several variations are designed and implemented. In our computational experiments, we have generated test instances based on real-world data. Experimental results show that the designed metaheuristic approaches provide high quality solutions compared to solutions obtained by the CPLEX solver. Further, one of the proposed GRASP variants has been adapted for the homogeneous VBPP and tested on standard benchmark instances in order to evaluate its performance against existing metaheuristic. The final conclusion is that the GRASP presents a promising approach for more challenging instances for which CPLEX cannot find feasible solution within a reasonable time limit.**

***Keywords*—Vector bin packing; container transport; GRASP; container packing problem.**

I. Introduction

In the second half of the 20th century there has been a gigantic increase in international trade. One of the main tools that made this possible is the containerization of the transport of goods, which started in the 1960s. The logistics behind such systems is highly complex. In case of the large scale logistics, the focus is on the optimization of operation at container terminals and marine-time shipping companies. On a smaller scale there is optimization of stowage plans of vessels and inland movement of containers by trucks and trains. An overview of related problems can be found in [1], [2]. An even lower level of optimizing container transport is on the level of a container itself, more precisely on which items will be paced inside a container. An overview of related problems can be seen in [3].

In this paper, we focus on solving problems related to container packing. Having in mind the costs of transport, an efficient container transport demands a highly dense packing in order to reduce the number of containers used for each delivery. The problem of packing packages in several types of containers arises in large transport systems, and can efficiently be represented using combinatorial optimization models. The typology of cutting and packing problems was introduced by Dyckhof in 1990 [4] and improved by Wäscher et al in 2007 ([5]). An overview of different variants of the packing problem can be seen in [4] and [5].

Basically, in packing problems, a set of packages should be placed in a set of containers. Each package is characterized by some measure and related value. For example, in this study, we considered weight and volume of packages. On the other hand, the type of container determines its limits for the total mass and volume, as well as the transportation cost while using it. This practical problem can be modeled using the homogeneous vector bin packing problem (H-VBPP). The H-VBPP is NP-hard optimization problem [6], that is defined as the problem of packing items in bins that have the same capacity and all other characteristics. The objective of this problem is to minimize the cost of packing all the items, while satisfying the problem constraints related to bin capacities. More about this problem can be founded in [7] and [8]. In the study by Caprara and Toth from 2001 [7], a library of 400 instances for the H-VBPP was published. Recently, in the work by Brandao and Pedroso in 2016 [9] all but 70 of them remained unsolved. In this paper we focus on solving the heterogeneous version of the VBPP, which is significantly less explored in literature, but we also used the data set from [7] to evaluate the adaptation of one of the proposed methods for the H-VBPP.

In the problem that we considered, the goal is to pack all packages into containers of different types with different limits of mass and volume while minimizing the total transportation costs. This problem is classified as the heterogeneous vector bin packing problem (Het-VBPP) and is proven to be NP-hard optimization problem in 1979 by Garey and Johnson [6]. The Het-VBPP with two-dimensional constraints was proposed in 1994 by Han *et al.* [10]. The problem is denoted as Multiple-

978-1-5386-6445-2/18/$31.00 ©2018 IEEE

type, two-dimensional bin packing problem and the authors applied simulated annealing to a randomly generated set of instances. The Het-VBPP has also been applied to the machine reassignment problem by Gabay and Zaourar (2016) [11]. The problem considered in [11] does not belong to the class of optimization problems, but rather to the class of decision problems, as the goal was to determinate if packing the set of packages is possible or not. The authors applied 34 greedy heuristic methods for solving the considered problem.

The Het-VBPP appears in many areas. Similar problems appear in industry and logistics in relation to loading of vehicles, ships and planes, database backup on removable media etc. [12]. For example, in the computer process scheduling there are several resources that need to be managed and they are limited. If the computers do not have the same characteristics, the problem is heterogeneous.

Metaheuristic methods are widely used for solving NP-hard optimization problems. They presents a family of approximate optimization techniques that became very popular in the past three decades [13]. A wide range of different metaheuristic methods have been successfully applied to a variaty of bin packing problems. For example, Mannai and Boulehmi [12] applied the guided tabu search metaheuristic for the homogeneous 2D vector bin packing problem. An iterative local search algorithm was designed for solving the same problem with additional constraints regarding the transportation cost by Hu *et al.* [14]. Genetic algorithms were developed for solving the H-VBPP in [15] and [16] . In addition, the variable neighborhood search metaheustics has been applied to the same problem in [17] and [18]. The same authors developed a memetic algorithm for this problem but with general costs in [19]. For solving the Het-VBPP, Han *et al*. [10] applied simulated annealing and Gabay and Zaourar [11] developed greedy heuristics.

In this study, we present a mathematical formulation for the Het-VBPP. Next, a randomized greedy algorithm is developed. With the goal of improving the quality of solutions acquired in this way, a local search is develeoped and used to extend the greedy algorithm to the GRASP metaheuristic. The GRASP is a simple multi-start metaheuristic introduced in [20], [21], that has been used for solving various optimization problems including different transportation problems. Besides the basic, many variants of GRASP were proposed in literature, such as Reactive GRASP [22], GRASP with Path Relinking [23] or Multiple Phase Neighborhood Search - GRASP [24]. In addition, GRASP is combined with different metaheuristics leading to efficient hybridization methods. An overview of different variants of the GRASP method and their applications can be found in [25]. Three variations of the GRASP are explored in this study.

The rest of the paper is organized as follows. In Section II a mathematical formulation of the considered problem is presented. The proposed metaheuristic solution approach is described in Section III. Experimental results are presented and analyzed in Section IV. In the final section, we give concluding remarks and potential directions for future work.

II. Mathematical Formulation

Starting from the mathematical formulation of a more general problem from [10], denoted as multi-type two-dimensional bin packing problem, we formulate an Integer Linear program (ILP) that adequately presents the Het-VBPP. In order to present the proposed Integer Linear Program (ILP) formulation, the following notation is introduced:

- nt - The number of container types;
- np - The total number of packages;
- C_t - Transportation costs of using a container of type t, defined for $t = 1..n$;
- LV_t - Volume limit for a container of type t, defined for $t = 1..n$;
- Lm_t - Mass limit for a container of type t, defined for $t = 1..n$;
- V_i - Volume of package i, defined for for $i = 1..np$;
- m_i - Mass of package i, defined for for $i = 1..np$;
- Ln_t - Limit in the number of containers of type t, defined for $t = 1..nt$.

The proposed model uses the following decision variables:

- p_{ijt} - is a binary variable that states if package i is placed in container j of type t. If this is true it is equal to 1, otherwise 0. It is defined for $i = 1..np$, $j = 1..Ln_t$ and $t = 1..nt$
- k_{jt} - is a binary variable that indicates if container j of type t is engaged. If this is true it is equal to 1, otherwise 0. It is defined for $j = 1..Ln_t$ and $t = 1..nt$

The objective function of the model is as follows

$$\min \qquad C = \sum_{t=1}^{nt} \sum_{j=1}^{Ln_t} k_{jt} \cdot C_t \tag{1}$$

The objective function (1) is used to minimize the total transportation costs. The used decision variables need to satisfy the following set of constraints:

$$\sum_{t=1}^{nt} \sum_{j=1}^{Ln_t} p_{ijt} = 1 \qquad i = 1..np \tag{2}$$

$$\sum_{i=1}^{np} p_{ijt} \cdot V_i \leq k_{jt} \cdot LV_t, \qquad t = 1..nt, j = 1..Ln_t \tag{3}$$

$$\sum_{i=1}^{np} p_{ijt} \cdot m_i \leq k_{jt} \cdot Lm_t, \qquad t = 1..nt, j = 1..Ln_t \tag{4}$$

The constraint (2) ensures that each package must be placed in exactly one container. Constraints given in Equation (3) and (4) guarantee that the limits of volume and mass of each container are not exceeded, respectively.

The proposed formulation is an adaptation of the formulations presented in [10] and [11]. In the study [10], the authors considered a similar problem, denoted as multi-type two-dimensional bin packing problem and presented a formulation using two dimensional bin capacities in a normalized form. In [11] an ILP formulaiton is given for the Het-VPBB, for

the case in which all the bins are unique. Note that in our model, we use non-normalized capacities (mass and volume) and multiple containers of the same type are allowed. In this way it is possible to more precisely model the corresponding real world problem.

III. GRASP

In this section we present the proposed variations of the GRASP metaheuristic for solving the Het-VBPP. We first present a constructive greedy algorithm and the used heuristic. Next, a simple local search is introduced. In the later part of this section a basic GRASP algorithm is presented. This algorithm is extended to two standardly used variations, Uniform GRASP (U-GRASP) and Reactive GRASP (R-GRASP).

In our approach for solving the Het-VBPP, the solution $S = (s, t)$ consists of an integer matrix s and an array t. The number of rows and columns of matrix s is np. Each row corresponds to one container and contains the indices of packages that are placed in it. We assume that each package can be placed in each type of container. The auxiliary array t of length np contains the information about the type of each container.

A. Greedy algorithm and Local search

1) Greedy algorithm: In this subsection, we present the greedy constructive algorithm. The greedy method will not be presented in the standard form of using a heuristic function for selection the element that will be used for expending the partial solution. Instead, we will define a candidate list from which an element will be selected semi-randomly for expansion of the partial solution. The reason for this is that the greedy constructive algorithm is used as a base for the GRASP, so we need a greedy randomized constructive (GRC) algorithm which can easily be defined in this way.

The main objective of the GRC is to construct a feasible solution by adding elements to the partial solution. The basic idea of the algorithm is to add new packages to containers that are already used, and if this is not possible to start filling additional ones. Obviously, it is much more difficult to fit packages having a large mass or volume in already used containers, so we prefer placing them at earlier stages. For this reason we use the following approach.

At the begining, we generate two candidate lists clm and clv that are initially equal to the sorted lists of packages SM and SV according to their mass and volume, respectively. All elements of matrix s are set to 0, as well as the number of packages that are placed in containers ($bp = 0$). After that, one by one row of s is filled in the following way. For each row that corresponds to a container, the type of container is chosen randomly. If the chosen container is of the first type, i.e., $t(i) = 1$, the candidate list $cl = clm$ is used, otherwise $cl = clv$. The purpose of an idea of using two candidate lists is to give priority to the packages of larger mass when packing in the container with larger mass limit, and to the packages of larger volume while placing them in the container with larger volume limit. For each element in the candidate list the incremental costs are defined and calculated in order to measure the changes that an element brings to objective function value, if included in the partial solution. The elements are than randomly chosen from the $restricted$ candidate list (rcl) according to the probability distribution. The restricted candidate list is a subset of candidate list, containing the elements that will increase the objective function value for relatively smaller amount when added to the partial solution. The size of the restricted candidate list is controlled by parameter $\alpha \in [0, 1]$. Each package k that has an incremental cost $ic(k) \in [c_{min}, c_{min} + \alpha(c_{max} - c_{min})]$ is included in rcl. In the case of Basic GRASP method, this parameter α has a fixed value [25].

After the package is assigned to the considered container, the number of packages that are placed (bp) is increased. In a case when the chosen package from rcl cannot fit in the considered container, the generation of solution S continues from the next container. GRC phase stops when all packages are placed in containers and bp reaches the total number of packages np, returning the feasible solution S. The pseudo code for the proposed method can be seen in Alg. 1.

Algorithm 1 Greedy randomized construction

```
procedure GRC(SM, SV, α)
    clm ← SM;
    clv ← SV;
    i ← 0; j ← 0; S ← 0; b_p ← 0;
    repeat
        if (j == 0) then
            Randomly select t(i) type of container i;
        if (t(i) == 1) then cl ← clm;
        else    cl ← clv;
        calculate ic_min from cl(1);
        calculate ic_max from cl(length(cl)));
        rcl ← ∅;
        for (each k ∈ cl) do
            if (ic_min ≤ ic(k) ≤ ic_min + α(ic_max − ic_min)) then
                rcl ← rcl ⋃ {k};
        Randomly select k from rcl;
        if (package k can be placed in container i) then
            s(i, j) ← k; j ← j + 1; b_p ← b_p + 1;
            Delete package k from both clm and clv;
        else
            if (j > 0) then                    //Next container
                i ← i + 1; j ← 0;
    until (b_p ≥ n_p)
    return S;
```

2) Local search: The neighborhood of the solution generated in GRC is searched within local search (LS) phase in order to improve the incumbent solution. The changes in a solution regarding the number of containers that are involved in transport can decrease the objective function value. Based on this idea, LS step in our implementation is designed as follows. In each container the packages are sorted in increase order according to their mass, as the volume of containers are relatively similar. In addition, the set of containers is sorted in increase order with respect to the total mass of all packages that are placed in each container. The LS starts from the

container with the lowest total mass of packages and tries to move its largest package to the last container in the sorted list of containers. If it is possible, having in mind the limits of mass and volume, the move is done and the procedure continues from the next package in its sorted list of packages. Otherwise, the algorithm tries to move the same package in the next container in the sorted list of containers. In a case when a package can not be moved to any other container, the observed container can not be emptied and the proposed LS undoes all previous moves while trying to unload the observed container. On the other hand, if a container is emptied, the improvement is achieved and the set of containers and packages in each of them are resorted as described. LS procedure stops when it passes the set of all containers without success in reducing the number of containers engaged in transport.

B. GRASP algorithms

1) Basic GRASP: The basic GRASP (B-GRASP) algorithm is an iterative procedure that includes two steps in each iteration: Greedy Randomizes Construction (GRC) and Local Search (LS) phase. These steps alternate until some stopping condition is satisfied.

The structure of the proposed B-GRASP is presented in Algorithm 2. In the preprocessing phase two sorted lists of packages SM and SV in decreasing order respecting their mass and volume, respectively, are generated. Incremental costs are calculated for each package in the sorted list as the reciprocal value to the value of its mass or volume, respectively for the two sorted lists SM and SV. The main idea was to give priority to the packages of larger mass or volume.

Algorithm 2 The Basic GRASP method

procedure BASIC GRASP($ProblemData, \alpha, t_{max}$)
  Create $SM = Sorted\ list\ of\ packages\ by\ mass$;
  Create $SV = Sorted\ list\ of\ packages\ by\ volume$;
  repeat
    $S \leftarrow GRC(SM, SV, \alpha)$; *//Solution construction*
    $S' \leftarrow LocalSearch(S)$: *//Local Search step*
    if $(f(S') < f(S))$ **then** *//Update solution*
      $S \leftarrow S'$;
  until $SessionTime \geq t_{max}$

2) Uniform GRASP: We implemented another variant of the GRASP metaheuristic, denoted as Uniform GRASP (U-GRASP). The only difference between U-GRASP and B-GRASP refers to the values of parameter α. Namely, in U-GRASP implementation, α is randomly chosen from the set $[0, 1/k]$ according to the uniform probability distribution, where k represents an integer. All other elements in the proposed U-GRASP implementation: solution representation, GRC and LS phases, as well as the termination criterion are the same as in B-GRASP. Algorithm 3 presents our U-GRASP implementation.

3) Reactive GRASP: As the proposed B-GRASP, Reactive GRASP has similar structure consisting of the two main steps: Greedy Randomizes Construction (GRC) and Local Search (LS) phase, that alternate until stopping condition is met. The common elements for all the three proposed variants of GRASP method are: solution representation, GRC and LS phases and the termination criterion.

Algorithm 3 The Uniform GRASP method

procedure UNIFORM GRASP($ProblemData, k, t_{max}$)
  Randomly choose $alpha \in [0, 1/k]$;
  Create $SM = Sorted\ list\ of\ packages\ by\ mass$;
  Create $SV = Sorted\ list\ of\ packages\ by\ volume$;
  repeat
    $S \leftarrow GRC(SM, SV, \alpha)$; *//Solution construction*
    $S' \leftarrow LocalSearch(S)$: *//Local Search step*
    if $(f(S') < f(S))$ **then** *//Update solution*
      $S \leftarrow S'$;
  until $SessionTime \geq t_{max}$

The main difference of the proposed R-GRASP compared to B-GRASP and U-GRASP is that it includes a learning mechanism while tuning the parameter α. Namely, the self-tuned parameter $\alpha \in [0, 1]$ is used to lied to the higher probabilities of choosing the promising candidates. The set of π values for α, i.e., $\alpha \in \{\alpha_1, \alpha_2, \ldots \alpha_\pi\}$. is used and these values have initially the same probabilities $p_i = 1/\pi$, $i = 1, \ldots, \pi$, that are updated according to the quality of the obtained solutions after every γ iterations. As it is proposed in [26], the average of the obtained objective values in all GRASP iterations while using a single value of α_i for parameter α is calculated and denoted by A_i. Using these values, the probabilities are being recalculated as:

$$p_i = \frac{q_i}{\sum_{j=1}^{m} q_j}, \quad q_i = \frac{z^*}{A_i},$$

where z^* presents the objective function value of the best found solution. In this way, the probability of those α_i, that lied to the higher quality solutions is being increased. The first π iterations are performed, one for each of the allowed values for parameter α, in order to set the initial value for A_i. The GRC phase of Reactive GRASP algorithm is returning feasible solution S. The Algorithm 4 describes the structure of the proposed Reactive GRASP for the considered problem.

C. The adaptation of the U-GRASP for H-VBPP

In this section, we show how the proposed GRASP can easily be adapted to the HVPP. As it will be seen in the results section, the best performing variant of the three GRASP methods on the Het-VBPP is U-GRASP. Due to this reason, we show how this variant is adapted to the H-VBPP. We considered our U-GRASP implementation restricted to only one type of container. More precise, in GRC phase of the adapted U-GRASP for H-VBPP the type of container is fixed instead of chosen randomly. In addition, we used both of candidate lists in GRC phase of the adapted U-GRASP, but as all containers are of the same type in this case, at the beginning of GRC phase, the algorithm chooses randomly one of the created candidate lists. Another difference, between the H-VBPP and the Het-VBPP, is the objective function calculation. Due to the to homogeneity of containers, the objective in

H-VBPP is usually to minimize the number of engaged containers in transportation, as homogeneous containers have the same transportation costs and the total transportation costs are proportional to the number of containers. Other steps of the adapted U-GRASP implementation remain the same as in U-GRASP for Het-VBPP.

Algorithm 4 The Reactive GRASP method

procedure REACTIVE GRASP($ProblemData, \gamma, t_{max}$)
  $n_{iter} \leftarrow 0$;
  Initialize probabilities;
  Create $SM = Sorted\ list\ of\ packages\ by\ mass$;
  Create $SV = Sorted\ list\ of\ packages\ by\ volume$;
  repeat
    $S \leftarrow GRC(SM, SV, \alpha)$; *//Solution construction*
    $S' \leftarrow LocalSearch(S)$: *//Local Search step*
    if $(f(S') < f(S))$ **then** *//Update solution*
      $S \leftarrow S'$;
    $n_{iter} \leftarrow n_{iter} + 1$;
    if $(n_{iter}\ mod\ \gamma = 0)$ **then**
      Update probabilities;
  until $SessionTime \geq t_{max}$

IV. EXPERIMENTAL RESULTS

In this section we present the results of the performed computational experiments. There objective was to evaluate the performance of the presented ILP and the variations of the GRASP metaheuristic. To be able to achieve this a wide range of test problem instances have been generated using real-world data. We observed the computational cost of each of the methods and the quality of found solutions

A. Experimental setup

All experimental results in this study are obtained using an Intel Core i7-2600 processor on 3.40GHz with 12GB RAM memory under Linux operating system. We have generated instances having a from 50 to a 1000 packages. The mass of each package is randomly generated integer value from $[1, 15]$ tons and its volume is randomly generated from $[1, 25]\ m^3$, both values are chosen by uniform probability distribution. These boundaries have been selected in such a way to make it possible for each package to be placed in each container.

The selection of container capacities and prices have been done based on real-world data. In the following text a short description of the three standard types of containers, defined by ISO 668:1995 standard ([27]), is given. According to this standard, the main characteristics of a container are: length, width and height, as well as allowed weight and volume. The width of each one of three types of container is the same, but their lengths can take the values of 6.058 and 12.192 meters, that correspond to the measures of 20 and 40 feet, respectively. The height of 40-feet containers is 2.591 or 2.896 meters (8.5 and 9.5 feet), while all 20-feet containers are 2.591m (8.5 feet) height. Therefore, we recognize three type of container: 20'-container, 40'-container and 40'-higher container. The interior volume, mass and capacity of an empty container are also standardized. In this study, the interior volume and the transport capacity of container are important characteristics, since they are directly related to the limits used in the problem. Table I presents the limits of mass and volume for the described three types of containers:

TABLE I
LIMITS OF MASS AND VOLUME FOR CONTAINERS

	20'-container	40'-container	40'-higher container
volume	$30m^3$	$60m^3$	$70m^3$
mass	$25,8t$	$24,5t$	$24,5t$

The transportation costs for containers described in Table IV-A are taken from the study Rajković et. al. (2015) [28]. These values correspond to the real-life transport costs of intermodal container transport from Shanghai to Belgrade, considered in [28]. The cost of transporting a container is: 1594 EUR, 2470 EUR i 2483 EUR, depending on the type. According to [28], these values present a real-life transport costs from Shanghai to Belgrade. We imposed the time limit for the designed metaheuristic methods as $t_{max} = 600$s for all tested instances. For each method 30 independent runs are performed on each instance and we present the average performance.

As a standard library of instances representing the Het-VBPP cannot be found, we applied the adapted U-GRASP method for the H-VBPP using 400 instances of H-VBPP, provided by Caprara and Toth (2001) [7]. This is done to evaluate its performance against existing metaheuristics. This library consists of 400 test instances that are divided in 10 classes denoted by $1, 2, \ldots, 10$. Each class includes 4 groups of 10 instances with the same number of items (that corresponds to the number of packages). The number of items within 4 groups of classes $1, 2, \ldots, 9$ are: 25, 50, 100, and 200, meaning that the first group includes 10 instances with 25 items, the second with 50 items etc. The class 10 consists of 4 groups with the following number of items: 24, 51, 99, 201. Another study by Brandao and Pedroso (2016) [9] considered the same set of instances and they provided optimal solutions for 330 of 400 instances. Before [9], the number of known optimal solutions on this data set was 212, and 188 of them were unsolved to optimality. The obtained solutions for all 330 instances can be found at https://research.fdabrandao.pt/research/vpsolver/results/2cbp.html#2cbp.

B. Results

First, we tested the proposed Integer Linear Programming formulation, presented in Section II, within CPLEX 12.6.2 solver and the imposed time limit of 600s. Table II presents the experimental results on generated set of instances obtained by CPLEX solver. Each instance is characterized by the total number of packages (np) that is contained in the first column. The next three columns refer to the results obtained by CPLEX solver and consist of the obtained upper and lower bounds (*UB* and (*LB*) for objective value and the CPU time ($t(s)$) in seconds that CPLEX needed to reach the obtained feasible solution.

TABLE II
SOLUTIONS OBTAINED BY CPLEX 12.6.2 SOLVER

np	$CPLEX$		
	UB	*LB*	$t(s)$
50	32393	30398.95	575.18
70	**47996**	45349.16	576.16
100	**65911**	62834.24	342.64
120	**68376**	65301.77	530.65
150	**88069**	84667.92	592.86
200	**125988**	118014.05	516.02
350	**221843**	206110.49	577.28
500	3273500	291891.06	137.55
750	4910250	439092.25	536.21
1000	/	/	600.00

As it can be seen from Table II CPLEX provided feasible solutions, that are not guaranteed optimal as the values for UB and LB are mutually different, for all tested instances except for the last one, with 1000 packages within the imposed time limit of 600s. Therefore, for this set of instances optimal solutions are not known. The best obtained results in this study are bolded, and, therefore, from Table II it can be concluded that CPLEX provided the best solutions on instances with 70, 100, 120, 150, 200 and 350 packages, but when it comes to instances of larger dimension as the last one with 1000 packages, CPLEX is not able to obtain even a feasible solution.

The proposed B-GRASP, U-GRASP and R-GRASP methods are tested on the same set of instances. Based on the preliminary parameter tuning tests, that are conducted on a subset of the described data set, the values for parameter α in B-GRASP, parameter k in U-GRASP, as well as parameters γ and π in R-GRASP method are determined. From the results of parameter tuning experiments, it was noted that high quality solutions can be obtained while using the small values for parameter α. The best results for fixed value of the parameter $\alpha \in [0, 1]$ were obtained with $\alpha = 0.05$, and therefore this value was used in all B-GRASP tests. At the beginning, 10 values of parameter α were considered from the following set $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. From the obtained results it was obvious that small values of parameter α lied to the high quality solutions. Therefore, we included the additional three values: 0.05, 0.15, and 0.25. Finally, the best results were reached with $\alpha = 0.05$. For the same reason parameter k in U-GRASP is set to 10, among all tested values from the set $\{1, 2, 3, \ldots, 10\}$ and parameter π in the proposed R-GRASP metaheuristic is set to 5 in order to include the 5 smallest allowed values of parameter α in our R-GRASP implementation:

$$\alpha \in \{0.05, 0.1, 0.15, 0.2, 0.25\}.$$

Finally, the value $\gamma = 100$ improved the quality of R-GRASP solutions. Each of the proposed three metaheuristics were run 30 times.

The objective function values of the best solutions obtained by the proposed B-GRASP, U-GRASP and R-GRASP methods are presented in Table III, that is organized as follows. The total number of packages for each instance is presented in the first column and the next three columns (*best*) contain the best solutions of the proposed B-GRASP, U-GRASP and R-GRASP, respectively, in 30 successive runs. The best values in each column are bolded.

TABLE III
THE OBJECTIVE VALUES OF BEST SOLUTIONS OBTAINED BY THE PROPOSED GRASP METHODS

np	*best*		
	$B-GRASP$	$U-GRASP$	$R-GRASP$
50	32380	**31543**	32380
70	48701	**47996**	48701
100	67585	66787	67637
120	71828	70102	70952
150	92410	91508	92397
200	131285	127965	132148
350	231224	225942	233642
500	330486	**323491**	328081
750	495737	**483445**	495467
1000	675209	**659034**	674973

TABLE IV
THE AVERAGE RUNNING TIMES THAT THE PROPOSED GRASP METHODS NEEDED TO REACH THEIR BEST SOLUTIONS

np	*t(s)*		
	$B-GRASP$	$U-GRASP$	$R-GRASP$
50	157.23	**125.93**	269.17
70	280.79	**226.28**	309.49
100	**234.18**	281.97	273.93
120	258.57	319.66	**258.06**
150	**268.35**	298.87	296.83
200	**248.83**	289.60	322.94
350	**259.41**	336.84	282.75
500	330.51	311.06	**305.13**
750	335.19	**276.73**	299.92
1000	**239.35**	273.81	312.74

From the results presented in Table III it can be concluded that the best solution quality is provided by U-GRASP as it outperformed B-GRASP and R-GRASP on each instance of the considered data set. As it can be seen from bolded results in Tables II and III, for instances with 50, 500, 750 and 1000 packages U-GRASP provided better quality solutions compared to CPLEX. In the case of instance with 70 packages, solutions obtained by the U-GRASP and CPLEX coincide.

Table IV presents the average running times that proposed methods needed to reach their best solutions in 30 successive runs. Table IV is organized in similar way as Table III. From the the results in Table IV, it can be concluded that B-GRASP is slightly faster than the remaining two methods. It needed less time to reach its best solutions in the case of instances with 100, 150, 200, 350 and 1000 packages. But overall there is no significant difference between the three graph variants. The values for average percentage gap of the obtained solutions from the best ones in 30 successive runs are presented in Table V.

From Table V it is obvious that B-GRASP is the most stable of the three proposed methods, as its average gap is less than 1% for all instances except two (with 150 and 350 packages).

TABLE V
THE AVERAGE PERCENTAGE GAP OF THE OBTAINED SOLUTIONS BY THE PROPOSED GRASP METHODS FROM THE BEST ONES

np	*agap (%)*		
	$B-GRASP$	$U-GRASP$	$R-GRASP$
50	**0.028**	2.587	0.047
70	**0.754**	1.413	1.251
100	**0.311**	1.064	0.864
120	**0.678**	1.863	2.458
150	1.169	**1.196**	1.728
200	0.964	2.110	**0.831**
350	1.154	1.215	**0.524**
500	0.762	**0.662**	1.760
750	**0.639**	1.415	1.335
1000	**0.484**	0.979	0.843

TABLE VI
THE NUMBER OF H-VBPP INSTANCES SOLVED TO OPTIMALITY BY THE PROPOSED ADAPTED U-GRASP METHOD

Class name	*The number of items*			
	25/24	50/51	100/99	200/199
1	10	10	0	0
2	10	10	8	5
3	10	10	9	5
4	9	**10**	**8**	0
5	10	**10**	**10**	0
6	10	8	0	0
7	9	6	3	0
8	10	10	10	10
9	10	10	0	0
10	5	0	0	0

R-GRASP obtained its best solutions with the average gap less than 1% only for 5 instance. The proposed U-GRASP mostly reached its best solutions with the average gap that are between 1% and 2%. This is probably due to the fixed value of parameter α that is included in the B-GRASP and therefore this method has pure diversification compared to the remaining two GRASP variants that, in addition, lied to the lowest solution quality in average (see Table III).

The adapted U-GRASP was tested on the set of 400 instances provided in [7]. We run the proposed method 30 times on each instance with the imposed time limit of 120s. The number of instances solved to optimality of each of the 4 groups within 10 classes of the given data sets by our adapted U-GRASP are presented in Table VI that is organized as follows. The name of each class of instances is given in the first column. The next 4 columns are related to the number of instances from each of the 4 groups within a class that are solved to optimality. We denoted each column by the corresponding number of items that are included in all instances belonging to the observed group. The first value (before the sign "/") presents the number of items from classes $1, 2, \ldots, 9$ and the next value (after the "/") refer to the number of items in instances belonging to the class 10.

From the results presented in Table VI it can be seen that the proposed adapted U-GRASP provided optimal solutions in the case of 245 out of 400 tested instances. The method proposed in [9] is more successful, as it provided optimal solutions for 330 instances on the same set of instances. The remaining 70 instances, that stayed unsolved til now, belong to the class 4 and 5 and in both cases include 50, 100, and 200 items, as well as to the class 9 with 200 items. As the Table VI shows, our adapted U-GRASP (see bolded values) provided 38 optimal solutions on the instances that were unsolved til now. Note that the solutions obtained by metaheuristic methods, as well as the proposed adapted U-GRASP method, are not guaranteed to be optimal. Therefore, we generated the formula, based on the one proposed in [11], to define a lower bound of a VBPP, equal to

$$l_\infty = \max\{\lceil \frac{\sum m_i}{Lm} \rceil, \lceil \frac{\sum V_i}{LV} \rceil\}.$$

Having in mind that this formula presents the maximum of the two ratio, the first one is between the total mass of packages that are placed in each container and the limit in mass of the corresponding container, and the second is similar but refers to the capacity instead of mass, this lower bound allows us to prove optimality. Namely, according to [11], the obtained solution is optimal if the lower bound is equal to the objective value of the heuristic solution. In the case of the solutions provided by the proposed adapted U-GRASP method, this condition was fulfilled and therefore, our solutions are proven to be optimal. In addition, we increased the value of the time limit from 120s to 600s and tested the proposed adapted U-GRASP on the remaining 155 instances that were not solved to optimality with the previous time limit. In this way we provided two more optimal solutions that are not known and optimal solutions for 5 instances that were already solved in literature.

V. CONCLUSION

In this study, the NP-hard problem of packing a set of packages in a set of containers with imposed limits of mass and volume capacity of each container is considered. The ILP formulation is presented and tested in the framework of CPLEX exact solver, that could not provide optimal solutions within the imposed time limit on generated set of instances. In addition, the three solution approaches are designed based on GRASP metaheuristic that correspond to: Basic GRASP, Uniform GRASP and Reactive GRASP method. The parameter tuning tests were performed to determine the values of the parameters included in the proposed implementations, and the final tests are conducted on the set of 10 generated problem instances. Experimental results show that the Uniform GRASP method outperformed the remaining two regarding the quality of the obtained solutions, while Basic GRASP was slightly faster and the most stable one, having in mind the average time needed for providing its best solutions and the average percentage gap of all solutions from its best one in 30 successive runs.

All of the three proposed metaheuristic methods present the promising approaches for the considered problem, as they significantly outperformed the use of CPLEX for solving the ILP, when it comes to the test examples with 500 and more

packages. For these more challenging instances, the obtained CPLEX results, if there are any within the imposed time limit, are practically useless. Therefore, there is an obvious need for designing heuristic methods while solving instances with 1000 and more packages, as CPLEX cannot provide feasible solutions within a reasonable execution time.

Future work may be conducted in several directions: to design new metaheuristic methods or to combine the proposed GRASP implementations with another metaheuristics in order to improve the results, to include real values of mass and volume of packages and/or a new problem constraints arising from practice in order to extend the problem to the more challenging one from the aspect of modeling and designing solution approaches etc.

REFERENCES

[1] T. E. Notteboom, "Container shipping and ports: an overview," *Review of network economics*, vol. 3, no. 2, pp. 86-106, 2004.

[2] D. Steenken, S. Voß and R. Stahlbock, "Container terminal operation and operations research-a classification and literature review," *OR spectrum*, vol. 26, no. 1, pp. 3–49, 2004.

[3] A. Bortfeldt and G. Wäscher, "Constraints in container loading–a state-of-the-art review," *European Journal of Operational Research*, vol. 229, no. 1, pp. 1–20, 2013.

[4] H. Dyckhoff, "A typology of cutting and packing problems," *European Journal of Operational Research*, vol. 44, no. 2, pp. 145–159, 1990.

[5] G. Wäscher, H. Haußner and H. Schumann, "An improved typology of cutting and packing problems," *European journal of operational research*, vol. 183, no. 3, pp. 1109–1130, 2007.

[6] M. R. Garey and D. S. Johnson, "Computers and intractability: a guide to np-completeness," 1979.

[7] A. Caprara and P. Toth, "Lower bounds and algorithms for the 2-dimensional vector packing problem lower bounds and algorithms for the 2-dimensional vector packing problem," *Discrete Applied Mathematics*, vol. 111, no. 3, pp. 231–262, 2001.

[8] F. C. Spieksma, "A branch-and-bound algorithm for the two-dimensional vector packing problem," *Computers & operations research*, vol. 21, no. 1, pp. 19–25, 1994.

[9] F. Brandao and J. P. Pedroso, "Bin packing and related problems: general arc-flow formulation with graph compression," *Computers & Operations Research*, vol. 69, pp. 56–67, 2016.

[10] B. T. Han, G. Diehr and J. S. Cook, "Multiple-type, two-dimensional bin packing problems: Applications and algorithms," *Annals of Operations Research*, vol. 50, no. 1, pp. 239–261, 1994.

[11] M. Gabay and S. Zaourar, "Vector bin packing with heterogeneous bins: application to the machine reassignment problem," *Annals of Operations Research*, vol. 242, no. 1, pp. 161–194, 2016.

[12] F. Mannai and M. Boulehmi, "A guided tabu search for the vector bin packing problem," *The 2018 International Conference of the African Federation of Operational Research Societies (AFROS 2018), Tunis, Tunisia*, 2018.

[13] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[14] Q. Hu, A. Lim and W. Zhu, "The two-dimensional vector packing problem with courier cost structure," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. 2013, pp. 212–221.

[15] M. Haouari and M. Serairi, "Heuristics for the variable sized bin-packing problem," *Computers & Operations Research*, vol. 36, no. 10, pp. 2877–2884, 2009.

[16] M. Quiroz-Castellanos, L. Cruz-Reyes, J. Torres-Jimenez, C. Gómez, H. J. F. Huacuja and A. C. Alvim, "A grouping genetic algorithm with controlled gene transmission for the bin packing problem," *Computers & Operations Research*, vol. 55, pp. 52–64, 2015.

[17] M. Buljubašić and M. Vasquez, "Consistent neighborhood search for one-dimensional bin packing and two-dimensional vector packing," *Computers & Operations Research*, vol. 76, pp. 12–21, 2016.

[18] K. Fleszar and C. Charalambous, "Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem," *European Journal of Operational Research*, vol. 210, no. 2, pp. 176–184, 2011.

[19] Q. Hu, L. Wei and A. Lim, "The two-dimensional vector packing problem with general costs," *Omega*, vol. 74, pp. 59–69, 2018.

[20] T. A. Feo and M. G. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations research letters*, vol. 8, no. 2, pp. 67–71, 1989.

[21] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *Journal of global optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[22] M. Boudia, M. A. O. Louly and C. Prins, "A reactive grasp and path relinking for a combined production–distribution problem," *Computers & Operations Research*, vol. 34, no. 11, pp. 3402–3419, 2007.

[23] M. G. Resendel and C. C. Ribeiro, "Grasp with path-relinking: Recent advances and applications," in *Metaheuristics: progress as real problem solvers*. 2005, pp. 29–63.

[24] Y. Marinakis, "Multiple phase neighborhood search-grasp for the capacitated vehicle routing problem," *Expert Systems with Applications*, vol. 39, no. 8, pp. 6807–6815, 2012.

[25] P. Festa and M. G. Resende, "An annotated bibliography of grasp–part i: Algorithms," *International Transactions in Operational Research*, vol. 16, no. 1, pp. 1–24, 2009.

[26] M. Prais and C. C. Ribeiro, "Reactive grasp: An application to a matrix decomposition problem in tdma traffic assignment," *INFORMS Journal on Computing*, vol. 12, no. 3, pp. 164–176, 2000.

[27] ISO, "ISO 668:1995 Series 1 freight containers – Classification, dimensions and ratings – AMENDMENT 2: 45' containers," International Organization for Standardization (ISO), Tech. Rep., 2005.

[28] R. Rajković, N. Zrnić, D. Stakić, A. Sedmak and S. Kirin, "An approach to determine optimal number of containers for cargo stacking in function of transportation cost," in *Proceedings - 6th International Symposium on Industrial Engineering - SIE 2015, 24th-25th September 2015, Belgrade, Serbia*, 2015, pp. 300–303.