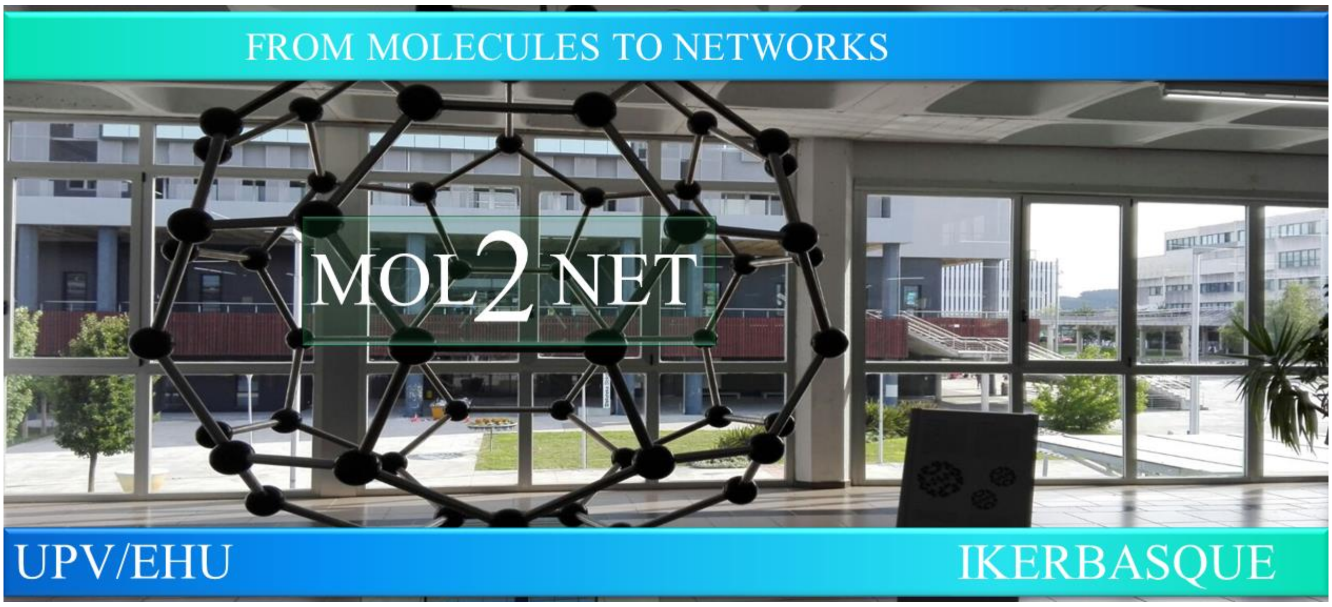
MOL2NET'21, Conference on Molecular, Biomedical & Computational Sciences and Engineering, 7th ed.



A rule-based arc-flow formulation for a generalized bin packing problem

Luis A. Gutierrez-Rodriguez [a], *Rogelio Jesús Corrales-Díaz* [a]

[a] Graduate Program in Systems Engineering, Nuevo Leon State University (UANL), Monterrey, Av. Universidad s/n, Col. Ciudad Universitaria, 66455 San Nicolas de los Garza, Nuevo Leon, Mexico.

luisgr_93@hotmail.com

Abstract

In the Generalized Bin Packing Problem (GBPP) different size containers can store objects and there is a cost per using the container. Each object is characterized by its volume and a benefit for being packed. The objects are subdivided into two groups, mandatory and non-mandatory. Mandatory items must be packed regardless their perk, while non-mandatory objects are optional to pack. In GBPP the smallest number of containers must be used, while non-mandatory items must be packed to increase the overall utility.

The arc-flow is an effective pseudo-polynomial formulation for Cutting and Packing problems. Here an edge represents each object, and the size of the object depends on the distance between the departure node and the arrival node. The resulting graph represents all possible combinations of objects and the way they can be located within a container.

In this work, we generate a rule-based digraph that introduces the loss arcs representing the empty space in the containers.

Keywords

Bin Packing, Generalized Bin Packing, Exact methods, Arc-flow formulation, Loss arcs, Rule-based Digraph.

1. Introduction

The Generalized Bin Packing Problem (GBPP) was introduced in [Baldi et al., 2012], as a variant of the Bin Packing Problem (BPP). In these problems each container has a capacity to store objects and a cost per use. Each object is characterized by a volume it occupies in the containers and benefit for being packed. The objects are subdivided into two groups: mandatory and non-mandatory. The mandatory objects must be packed regardless of their perk. The non-mandatory objects are optional to pack into containers. The objective in GBPP is minimizing the number of containers used, and the non-mandatory objects are packed to get a higher benefit and increase utility. The objective of the GBPP is closely related to two classic operations research problems, the Bin Packing Problem (packing objects in containers) and the Knapsack Problem (select to pack the most profitable objects).

Arc-Flow model is an effective pseudo-polynomial formulation for BPP. It was proposed in [Valerio De Carvalho, 1999], where a branch-and-price algorithm was used. In this formulation each object is represented by an edge, the size of the object depends on the distance that separates the departure and the arrival node. The resulting digraph represents all possible combinations of objects and the way they can be sorted into a container. In [Valerio de Carvalho, 2002] it was proposed a generalization of this model by allowing different container sizes on the same graph. Based on these works [Brandao and Pedroso, 2016] proposed a more efficient compression of the graph and is known as the General Arc-Flow Model (GAFM).

In this paper we propose another graph generation procedure similar to that used in [Valerio de Carvalho, 2002] but with rules to build a graph without symmetry add complement arcs called "loss arcs" to facilitate optimal solution for problem instances with many objects.

2. Materials and Methods

In this section we propose a variant of [Valerio de Carvalho, 2002] algorithm to build the asymmetric reduced graph. First the Arc-Flow Model is introduced and compared with similar known models, then the rules to build the digraph are considered.

2.1 The Arc-Flow Formulation

In the Arc-Flow formulation, we consider the vertices in the graph to represent the integer capacity of the container, from zero to maximum capacity. For example, if we have a container with a capacity of 7, we have 8 vertices $\{0, 1, \dots, 7\}$. An arc (i, j) represents an object having size $(j - i)$.

Let T be a set of containers with index t used to indicate container's type. It is assumed that containers are arranged according to their capacity in the decreasing order. Let K be a set of all objects to pack and O_k be a subset of K with the non-mandatory objects characterized by index k. Let C_t be a capacity of the container of type t. Let w_k be a volume of the object k^{th} type, p_k a profit of packing k^{th} type objects. Let b_k be the demand, the number of objects to be packed, of mandatory objects of type k. Denote by A^x a set of arcs which represent objects and let A^y be a set of loss arcs representing an empty space in the container. Finally, let $G = (V, A^x \cup A^y)$ be a digraph associated with GBPP.

2.1.1 Decision variables

$$\mathbf{z_t} = \textit{number of containers of type } \boldsymbol{t}$$
$$\boldsymbol{x_{de}} = \textit{number of arcs with size } \boldsymbol{e-d} \textit{ representing objects}$$
$$\boldsymbol{y_{de}} = \textit{number of arcs with size } \boldsymbol{e-d} \textit{ representing empty space}$$
$$\boldsymbol{e_i} = \begin{cases} \mathbf{1}, & \textit{if non-mandatory object } \boldsymbol{i} \in \boldsymbol{O_k} \textit{ is packed} \\ \mathbf{0}, & \textit{otherwise} \end{cases}$$

2.1.2 Arc-Flow Model with Loss Arcs

$$\min \sum_{t\in T} c_t z_t - \sum_{k\in K}\sum_{i\in O_k} p_k e_i \tag{1.1}$$

subject to

$$\sum_{(d,e)\in A^x} x_{de} + \sum_{(d,e)\in A^y} y_{de} - \sum_{(e,f)\in A^x} x_{ef} - \sum_{(e,f)\in A^y} y_{ef} = \begin{cases} z_t, e = C_t \\ -\sum_{t\in T} z_t, e = 0 \\ 0, e \neq 0 \end{cases} \tag{1.2}$$

$$\sum_{(d,e)\in A^x} x_{de} = b_k + \sum_{i\in O_k} e_i \quad \forall k \in K \tag{1.3}$$

$$z_t \leq U_t \quad \forall t \in T \tag{1.4}$$

$$\sum_{(v,C_t)\in A^y} y_{v,C_t} \leq z_t \quad \forall t \in T \tag{1.5}$$

$$e_i \in \{0,1\} \quad \forall i \in O_k \tag{1.6}$$

$$x_{de} \in \mathbb{Z}^+ \quad \forall (d,e) \in A^x \tag{1.7}$$

$$y_{de} \in \mathbb{Z}^+ \quad \forall (d,e) \in A^y \tag{1.8}$$

The objective (1.1) represents the utility defined as the difference between the cost of using the containers and the benefit of packing the non-mandatory objects. Constraints (1.2) state the flow conservation. The demand is defined as the sum of the demand for obligatory objects and the sum of the demand of the non-mandatory objects as stated in (1.3), In contrast to GAFM, where the demand for the objects is integer, in our model a decision is made for each object and corresponding binary variables are used. Constraints (1.4) state that for each type of containers, the solution is bounded by the number of available containers. The constraint (1.5) assures that the number of empty edges reaching a terminal vertex cannot exceed the number of edges that reach the terminal vertex. This helps maintaining the flow conservation for the arcs of empty spaces.

2.2 The Rule-Based Digraph

The digraph is constructed under the following assumptions:

1. The number of vertices is equal to the capacity of the largest container
2. A list of unique sizes of the objects is known.
3. For all objects, mandatory and non-mandatory, a demand for the objects is known.

Based on this information the following algorithm is proposed. More details on algorithms for graph construction are given in [Gutierrez-Rodriguez, 2019].

Algorithm 1: Build of digraph for GBPP

Input: ***m*** → the demand of the objects per size; ***l*** → list of unique size of the objects; ***C*** → list of unique capacities of the containers

Output: **V**→ set of Vertices; **A** → set of Arcs

```
1  Function buildDigraphGBPP (m, l, C): is
2  L ← max {c_i: i ∈ C};
3  V ← {0, …, L};
4  A ← ∅;
5  A^x ← ∅;
6  A^y ← ∅;
7  For t in l do:
8  |    A^x ← A^x ∪ {(0, t)};
9  Foreach u in V do:
10 |    T ← ∅;
11 |    mla ← 0;                                   #mla means Max Length Arc
12 |    Foreach (i, j) in A do:
13 |    |    If u = j and j-i > mla then mla ← j-i;
14 |    For k = 1 to m do:
15 |    |    If l_k ≤ mla then: T ← T ∪ {(u + l_k)};
16 |    Foreach v in T do:
17 |    |    If v ≤ L then: A^x ← A^x ∪ {(u, v)};
18 |    Foreach c in C do:
19 |    |    A^y ← A^y ∪ {(u, c)};
20  A ← A^x ∪ A^y;
21 Return G = (V, A);                             #Digraph without symmetry
```

Table 1. *Algorithm to build the asymmetric digraph for GBPP.*

2.2.1 Graph example

To show the graphs that we build the following example are used. Consider a set of containers of size 4 and 7. In which we can only pack objects of size 6, 3 and 2. In Figure 1 we show the build of digraph based in Valerio de Carvalho's method.

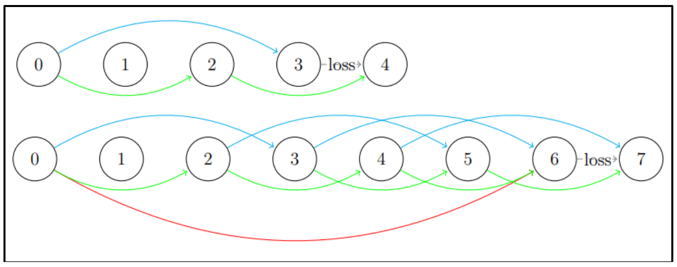


Figure 1. *A build foreach types of containers*

In the Figure 2 we compress and insert each lower capacity container on the larger container. We consider the nodes 4 and 7 nodes as final nodes for a Maximum flow problem. All solutions with z_2 are the quantity of containers with z_2 sized was occupied. Similarity, for each z_n where z_{n+1} is a lower capacity container than z_n.

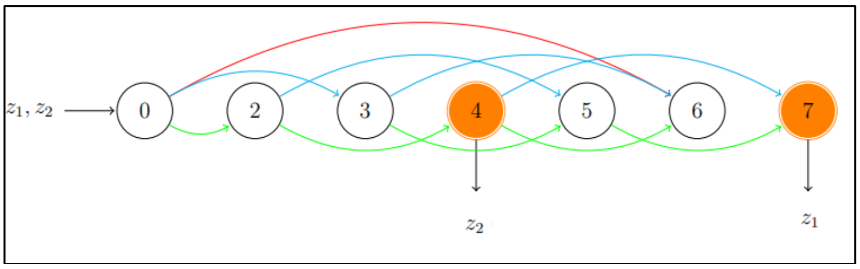


Figure 2. *A digraph equivalent of two graphs in Figure 1*

In Figure 3 we add the loss arcs, which represent the empty spaces in the container. The loss arc can arrive only to ***n*** node if these nodes are terminal nodes, which it means, is a specific capacity of some container.

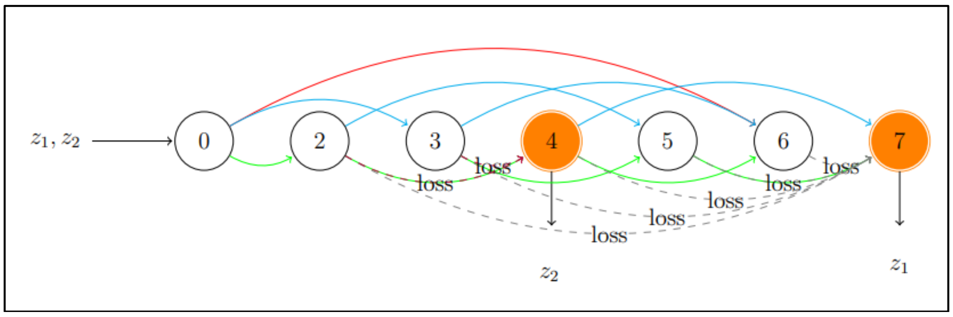


Figure 3. *A digraph for GBPP with loss arcs*

2.3 Methodology

In this paper, we compare the GAFM of [Brandao and Pedroso, 2016] vs the rule-based arc-flow model proposed in this paper. In both cases, the instances generated by [Baldi et al., 2012] were used for comparison. The set of instances available for GBPP is segmented into 4 classes.

Class 0: There are 300 instances that were originally created for the Variable Cost Sized Bin Packing Problem (VCSBPP) [Crainic et al., 2011], there are 10 instances for each possible combination of the following 4 parameters:

- Number of objects: 25, 50, 100, 200 and 500.
- Volume of objects: I1:[1, 100]; I2:[20, 100]; I3:[50, 100]
- Profit of objects: 200 for everyone, all mandatory.
- Container type:
    - 3 types: 100, 120 and 150
    - 5 types: 60, 80, 100, 120 and 150

The cost per use is equal to the capacity of the container. For each type of container there is a lower bound and an upper bound, the lower bounds are 0 and the bounds are given $\lceil V_{TOT} / V_t \rceil$ where V_{TOT} is the total volume of the objects, and V_t is the capacity of the type t container.

Class 1: These are the same instances as Class 0, but all objects are not mandatory, and the benefits of packaging the objects is given a distribution uniform $p_i \in \lceil \mathbb{N}(0.5,3) w_i \rceil$

Class 2: These are the same instances as Class 0, but all objects are Non-mandatory, and the benefits of packaging the objects is given a distribution uniform $p_i \in \lceil \mathbb{N}(0.5,4) w_i \rceil$

Class 3: These are a selection of 12 large instances (500 objects) of class 1 and class 2, with a representative mix of characteristics in terms of volume of items, item perks, item gains, and container types. For each instance, five instances were randomly obtained with 0%, 25%, 50%, 75% and 100% required items, for a total of 60 instances.

3 Results and Discussion

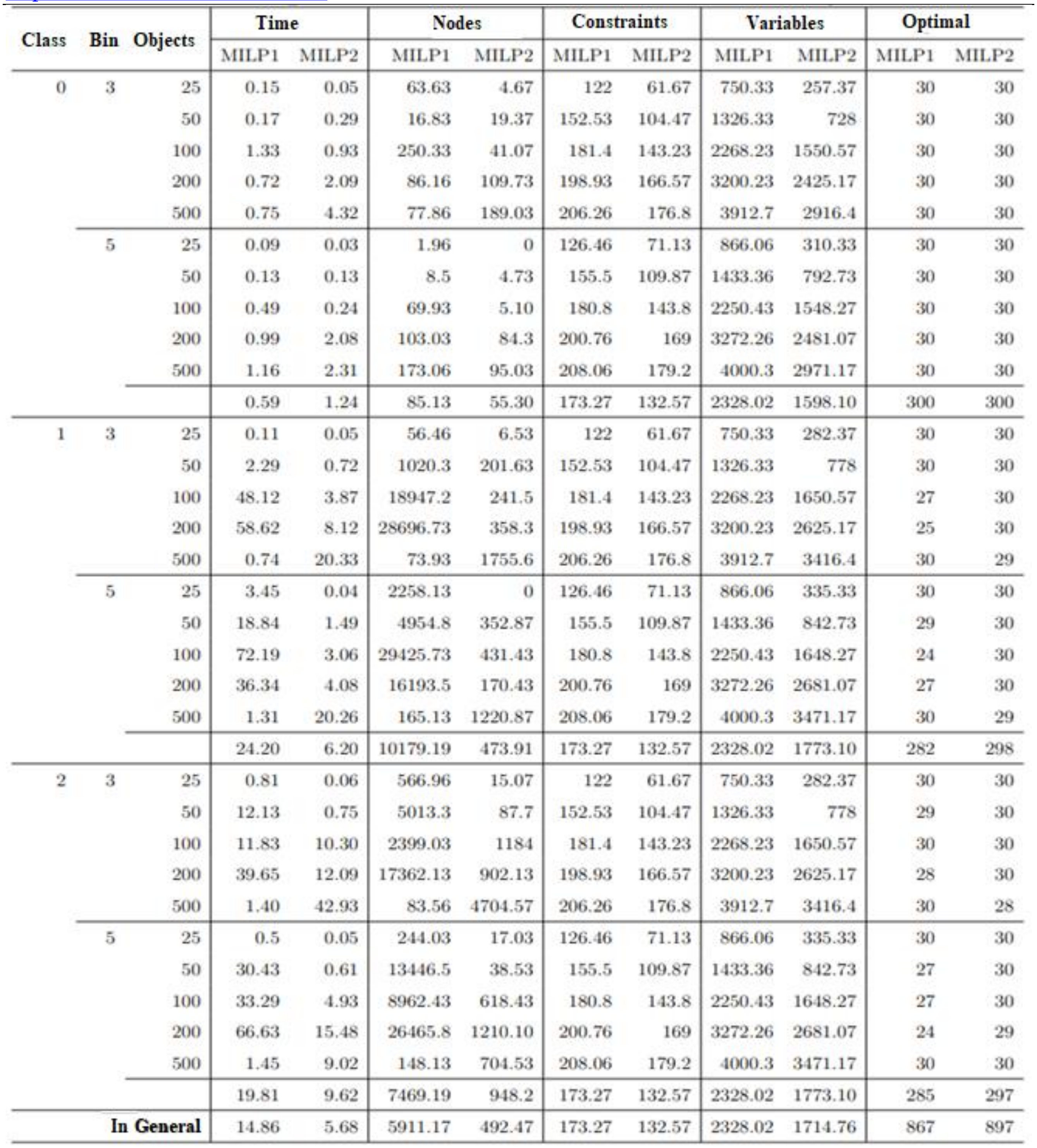
In this section we compare the results obtained by the rule-based arc-flow formulation for the generalized bin packing problem ($MILP_1$) vs the results obtained with GAFM ($MILP_2$)

3.1 Experimental environment

For the rule-based arc-flow formulation the experiment was carried out on an HP Z230 Workstation, which has an Intel Xeon CPU E3-1245 v3 @ 3.40GHz x 8 processor and 15.4 GiB RAM, the used version of CPLEX was 12.9.0. In CPLEX processing was limited to one thread, the time limit for the execution of solvers was 300 seconds. For the results obtained in [Brandao, 2017] use a computer with an Intel Xeon 2.66GHz Quad-Core processor was used. Mac OS X 10.11.6 operating system, with 16GB of RAM. The algorithm to generate the graph was developed in C++, the models were built in Python 2.7, and it was solved in Gurobi 7.0.2

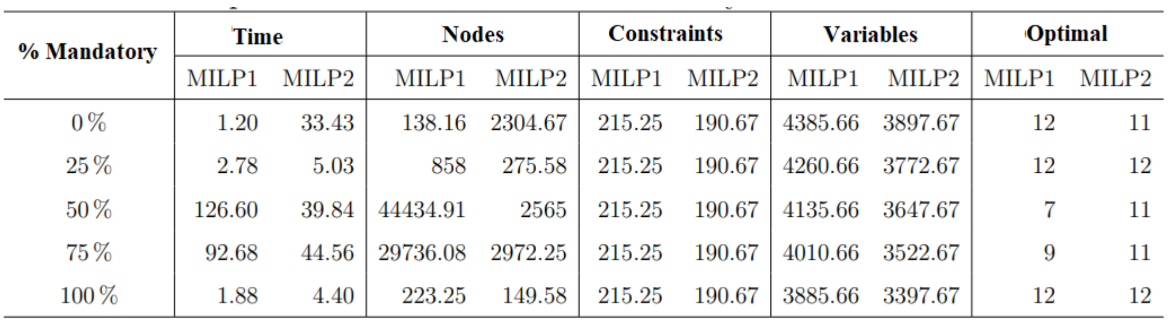
3.2 Results

In Table 2, the Time column indicates the resolution time; The Nodes column indicates the number of nodes explored by the CPlex branch-and-bound in $MILP_1$ and Gurobi in $MILP_2$; The Constraints column indicates the number of average constraints in each model; The Variables column indicates the number of average variables in each model; The Optimal column indicates the number of instances in which optimal results were obtained.



Class	Bin	Objects	Time		Nodes		Constraints		Variables		Optimal	
			MILP1	MILP2	MILP1	MILP2	MILP1	MILP2	MILP1	MILP2	MILP1	MILP2
0	3	25	0.15	0.05	63.63	4.67	122	61.67	750.33	257.37	30	30
		50	0.17	0.29	16.83	19.37	152.53	104.47	1326.33	728	30	30
		100	1.33	0.93	250.33	41.07	181.4	143.23	2268.23	1550.57	30	30
		200	0.72	2.09	86.16	109.73	198.93	166.57	3200.23	2425.17	30	30
		500	0.75	4.32	77.86	189.03	206.26	176.8	3912.7	2916.4	30	30
	5	25	0.09	0.03	1.96	0	126.46	71.13	866.06	310.33	30	30
		50	0.13	0.13	8.5	4.73	155.5	109.87	1433.36	792.73	30	30
		100	0.49	0.24	69.93	5.10	180.8	143.8	2250.43	1548.27	30	30
		200	0.99	2.08	103.03	84.3	200.76	169	3272.26	2481.07	30	30
		500	1.16	2.31	173.06	95.03	208.06	179.2	4000.3	2971.17	30	30
			0.59	1.24	85.13	55.30	173.27	132.57	2328.02	1598.10	300	300
1	3	25	0.11	0.05	56.46	6.53	122	61.67	750.33	282.37	30	30
		50	2.29	0.72	1020.3	201.63	152.53	104.47	1326.33	778	30	30
		100	48.12	3.87	18947.2	241.5	181.4	143.23	2268.23	1650.57	27	30
		200	58.62	8.12	28696.73	358.3	198.93	166.57	3200.23	2625.17	25	30
		500	0.74	20.33	73.93	1755.6	206.26	176.8	3912.7	3416.4	30	29
	5	25	3.45	0.04	2258.13	0	126.46	71.13	866.06	335.33	30	30
		50	18.84	1.49	4954.8	352.87	155.5	109.87	1433.36	842.73	29	30
		100	72.19	3.06	29425.73	431.43	180.8	143.8	2250.43	1648.27	24	30
		200	36.34	4.08	16193.5	170.43	200.76	169	3272.26	2681.07	27	30
		500	1.31	20.26	165.13	1220.87	208.06	179.2	4000.3	3471.17	30	29
			24.20	6.20	10179.19	473.91	173.27	132.57	2328.02	1773.10	282	298
2	3	25	0.81	0.06	566.96	15.07	122	61.67	750.33	282.37	30	30
		50	12.13	0.75	5013.3	87.7	152.53	104.47	1326.33	778	29	30
		100	11.83	10.30	2399.03	1184	181.4	143.23	2268.23	1650.57	30	30
		200	39.65	12.09	17362.13	902.13	198.93	166.57	3200.23	2625.17	28	30
		500	1.40	42.93	83.56	4704.57	206.26	176.8	3912.7	3416.4	30	28
	5	25	0.5	0.05	244.03	17.03	126.46	71.13	866.06	335.33	30	30
		50	30.43	0.61	13446.5	38.53	155.5	109.87	1433.36	842.73	27	30
		100	33.29	4.93	8962.43	618.43	180.8	143.8	2250.43	1648.27	27	30
		200	66.63	15.48	26465.8	1210.10	200.76	169	3272.26	2681.07	24	29
		500	1.45	9.02	148.13	704.53	208.06	179.2	4000.3	3471.17	30	30
			19.81	9.62	7469.19	948.2	173.27	132.57	2328.02	1773.10	285	297
In General			14.86	5.68	5911.17	492.47	173.27	132.57	2328.02	1714.76	867	897

Table 2. *Compare results between MILP$_1$ and MILP$_2$ on Class 0,1 and 2.*



% Mandatory	Time		Nodes		Constraints		Variables		Optimal	
	MILP1	MILP2	MILP1	MILP2	MILP1	MILP2	MILP1	MILP2	MILP1	MILP2
0 %	1.20	33.43	138.16	2304.67	215.25	190.67	4385.66	3897.67	12	11
25 %	2.78	5.03	858	275.58	215.25	190.67	4260.66	3772.67	12	12
50 %	126.60	39.84	44434.91	2565	215.25	190.67	4135.66	3647.67	7	11
75 %	92.68	44.56	29736.08	2972.25	215.25	190.67	4010.66	3522.67	9	11
100 %	1.88	4.40	223.25	149.58	215.25	190.67	3885.66	3397.67	12	12
In General	45.08	25.45	15078.08	1653.41	215.25	190.67	4135.66	3647.67	52	57

Table 3. *Compare results between MILP$_1$ and MILP$_2$ on Class 3.*

Table 3 has the same columns as Table 2, and it contains shows the comparison of the models on the instances of Class 3. We can note that in instances where the percentage of mandatory objects is 0%, 25% and 100% the processing times of $MILP_1$ are less than those of $MILP_2$, although in general $MILP_2$ takes less on average. Another point to note is that on average the $MILP_1$ model has fewer variables than the $MILP_2$.

3.3 Discussion

The main difference between Valerio de Carvalho's model and our model is considering a set of loss arcs to represent empty spaces in containers. Valerio de Carvalho's model also has the loss, but when reducing the graph, the loss arcs are removed. In the practice we consider "imaginary" objects and spaces that would be occupied by some objects of this size. That is the reason for having the demand constraint always greater than the real demand value. The objective in BPP is to pack all items in containers but, in GBPP the demand can be estimated exactly since you can depreciate some objects which is not profitable to pack. If we add a set of loss arcs, we add to the solver the capability of selecting an empty space before an object where the profit is lower than no pack it.

Although the models were not run on the same computer, both models manage to find many optimal values for instances of literature, within a reasonable time. However, it is noted that the method of compression of $MILP_2$ allows a significant reduction in the number of variables and restrictions resulting in less processing time. The $MILP_1$ model behaved almost as efficiently as the best model in literature.

Note is that the $MILP_1$ model finds solutions optimal in 500 object instances long before $MILP_2$, needs to be reviewed the structure of the models, to confirm this peculiar situation. a possible explanation is that $MILP_2$ in its compression algorithm discards paths in the for which there is not enough demand for small objects, for which reason these objects are sought for compatibility with other larger ones. In $MILP_1$ the edges of empty space to make these paths feasible and preserve the flow with these edges zero weight. Another possible explanation would be that being a large quantity of objects, these small objects in a group are enough to fill a container for them themselves, without having to look for compatibility and without having to belong to a path that could be discarded by the GAFM compression algorithms.

4. Conclusions

The rule-based arc-flow is a pseudo-polynomial formulation and is one of the best at solving the variants of the BPP of one dimension and the VCSBPP. We developed a method based on the network flow model for VCSBPP and check that it has good results also for the GBPP variant presenting results almost as good as the best-known network flow-based model, the GAFM, proposed by [Brandao, 2017].

An interesting direction for the future work is generating the GAFM vs the model proposed in this work by comparing it with most of the largest instances with 1000, 2000 and 5000 objects and adding to compare different types and sizes of containers.

References

[Baldi et al., 2012] Baldi, M. M., Crainic, T. G., Perboli, G. & Tadei, R. (2012). The generalized bin packing problem. *Transportation Research Part E: Logistics and Transportation Review, 48(6)*, 1205-1220. https://doi.org/10.1016/j.tre.2012.06.005

[Brandao and Pedroso, 2016] Brandão, F. & Pedroso, J. P. (2016). Bin packing and related problems: General arc-flow formulation with graph compression. *Computers & Operations Research, 69*, 56-67. https://doi.org/10.1016/j.cor.2015.11.009

[Brandao, 2017] Brandao, F. D. A. (2017). Cutting & packing problems: general arc-flow formulation with graph compression. *(Doctoral thesis, Universidade do Porto)*. https://repositorio-aberto.up.pt/handle/10216/110172

[Crainic et al., 2011] Crainic, T. G., Perboli, G., Rei, W. & Tadei, R. (2011). Efficient lower bounds and heuristics for the variable cost and size bin packing problem. *Computers & Operations Research, 38(11)*, 1474-1482. https://doi.org/10.1016/j.cor.2011.01.001

[Gutierrez-Rodriguez, 2019] Gutiérrez-Rodríguez, L. Á. (2019). Problema generalizado del empaquetamiento de contenedores: una comparación entre diferentes métodos de solución *(Master's thesis, Universidad Autónoma de Nuevo León)*. http://eprints.uanl.mx/id/eprint/17851

[Valerio De Carvalho, 1999] Valério de Carvalho, J. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, *86*, 629-659. https://doi.org/10.1023/a:1018952112615

[Valerio de Carvalho, 2002] Valério de Carvalho, J. (2002). LP models for bin packing and cutting stock problems. *European Journal of Operational Research, 141(2)*, 253-273. https://doi.org/10.1016/s0377-2217(02)00124-8