ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
"Antonio Ruberti"
CONSIGLIO NAZIONALE DELLE RICERCHE

T. Bacci, S. Nicoloso

A HEURISTIC ALGORITHM FOR THE BIN PACKING PROBLEM WITH CONFLICTS ON INTERVAL GRAPHS

R. 3, 2018

Tiziano Bacci – IASI - CNR — Via dei Taurini 19, 00185 Roma, Italia (tiziano.bacci@iasi.cnr.it).

Sara Nicoloso – IASI - CNR — Via dei Taurini 19, 00185 Roma, Italia (sara.nicoloso@iasi.cnr.it).

ISSN: 1128–3378

Collana dei Rapporti dell'Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", CNR

via dei Taurini 19, 00185 ROMA, Italy

tel. ++39-06-49937101/02
fax ++39-06-49937106
email: iasi@iasi.cnr.it
URL: http://www.iasi.cnr.it

Abstract

In this paper we deal with the Bin Packing Problem with Conflicts on interval graphs: given an interval graph, a nonnegative integer weight for each vertex, and a bound, find a partition of the vertex set of the graph into the minimum number of subsets such that the sum of the weights of the vertices assigned to the same subset does not exceed the bound and two vertices connected by an edge do not belong to the same subset. We design a heuristic algorithm and propose a new random interval graph generator which builds interval conflict graphs with desired edge density. We test the algorithm on a huge test bed and compare the results with the best heuristic algorithms: the experiments show that our method outperform the other ones when the average number of vertices per subset is greater than or equal to three.

Key words: Bin Packing with Conflicts, Interval Graphs, Threshold Graphs, Random Interval Graph Generator

1. Introduction

The *Bin Packing Problem with Conflicts* ($BPPC$), first introduced in a scheduling context (Jansen and Öhring (1997)), is defined as follows. Given a graph G, a nonnegative integer weight for each vertex, and a nonnegative integer B, find a partition of the vertex set of the graph into the minimum number of subsets such that two vertices connected by an edge do not belong to the same subset and the sum of the weights of the vertices assigned to the same subset is less than or equal to B.

$BPPC$ is a difficult combinatorial optimization problem largely studied in the literature. It arises in many applicative contexts, for example in Christofides et al. (1979) where some flammable, explosive, or toxic substances cannot be placed in the same vehicle.

As far as we know, all the heuristic algorithms proposed in the literature are designed for arbitrary graphs. If the conflict graph is an interval graph, one can take advantage of its peculiar properties in order to find good solutions for the problem. For this reason, we focus on $BPPC$ where the graph G is an interval graph and we propose a heuristic algorithm.

We present extensive computational experiments, comparing our algorithm with the best heuristic methods in the literature. To our knowledge, no tests on instances of $BPPC$ with arbitrary interval conflict graphs were performed in the literature. Since the existing random interval graph generators output graphs with edge density in a very narrow range, we design a new random interval graph generator which outputs interval graphs with desired edge density. We tested our algorithm also on the benchmark instances by Fernandes Muritiba et al. (2010), as Sadykov and Vanderbeck (2013) realized that their conflict graphs are interval graphs, and not arbitrary ones. Actually, the conflict graphs of these instances are not arbitrary interval graphs, but special ones, namely threshold graphs (see Section 6.1).

The paper is organized as follows. A formal definition of the problem can be found in Section 2. In Section 3, we review the literature on $BPPC$. In Section 4, we present our heuristic algorithm, while the random interval graph generator can be found in Section 5. Experimental results are discussed in Section 6 where we test our algorithm over literature instances (Section 6.1) an on instances with arbitrary interval conflict graphs (Section 6.2). Section 7 concludes.

2. Problem definition

The Bin Packing Problem with Conflicts ($BPPC$) is defined as follows. Given a graph $G = (V, E)$ with $n = |V|$ vertices, an integer weight $w_i \geq 0$ for $i = 1, \ldots, n$, and an integer $B \geq 0$, find a partition of V into k subsets $V_1, \ldots, V_k$, such that $\sum_{i \in V_j} w_i \leq B$ for $j = 1, \ldots, k$, two vertices connected by an edge do not belong to the same subset, and k is minimum. Such minimum value of k will be denoted k_{BPPC}. The graph $G = (V, E)$ is called *conflict graph* and two vertices connected by an edge are said to be *in conflict*. $BPPC$ generalizes two well known combinatorial optimization problems, the Bin Packing problem and the Vertex Coloring problem, which we now introduce.

The Bin Packing problem (BP) is defined as follows. Given a set V of n items, an integer weight $w_i \geq 0$ for $i = 1, \ldots, n$, and an integer $B \geq 0$, find a partition of V into k subsets (*bins*) $V_1, \ldots, V_k$, such that $\sum_{i \in V_j} w_i \leq B$ for $j = 1, \ldots, k$ and k is minimum. Such minimum value of k will be denoted k_{BP}.

The Vertex Coloring problem (VC) is defined as follows. Given a graph $G = (V, E)$ with $n = |V|$ vertices, find a partition of V into k subsets (*colors*) $V_1, \ldots, V_k$, such that two vertices connected by an edge do not belong to the same subset and k is minimum. Throughout the paper *h-coloring* denotes a feasible vertex coloring with h colors. The minimum value of h such that G admits a h-coloring is called the *chromatic number* $\chi(G)$ of the graph G. Notice that each V_i is an independent set, as well as each subset of a feasible solution of $BPPC$.

$BPPC$ generalizes both BP and VC. In fact, it coincides with BP when the edge set E of the graph G is empty, and it reduces to VC when $B \geq \sum_{i \in V} w_i$. Hence, since VC on arbitrary

graphs and BP are both NP-hard (Garey and Johnson (1978)), by generalization $BPPC$ is NP-hard too.

Given a $BPPC$ instance, the *underlying* BP is obtained from it by removing the conflict graph G and the *underlying* VC is obtained by ignoring weights and B. Considering these underlying problems, clearly $k_{BPPC} \geq \max\{k_{BP}, \chi(G)\}$.

In this paper we focus on $BPPC$ where the conflict graph $G = (V, E)$ is an interval graph. $G = (V, E)$ is an interval graph if every vertex $p \in V$ can be put in one-to-one correspondence with an open interval $I_p = (l_p, r_p)$ of the real line, and two vertices $p, q \in V$ are connected by edge $(p, q) \in E$ if and only if the corresponding intervals intersect, i.e. $l_p < r_q$ and $l_q < r_p$. The family of intervals $\mathcal{I} = \{I_h = (l_h, r_h), h = 1, \ldots, n\}$ is called an *interval model* for G. Any interval graph admits an interval model, and viceversa. Notice that when the edge set of the graph is empty, any set of n mutually non-intersecting intervals is an interval model for G. In what follows, w.l.o.g. we will assume that $l_j, r_j \in \mathbb{Z}_+$ for $j = 1, \ldots, n$ and that $\min\{l_j, j = 1, \ldots, n\} = 0$; we also define $R = \max\{r_j, j = 1. \ldots, n\}$. On interval graphs VC, hence $\chi(G)$, can be computed in linear time (Golumbic (1980)), nevertheless $BPPC$ with an interval conflict graph remains NP-hard. We observe that $\chi(G)$ is equal to the size $\omega(G)$ of a maximum-sized subset of mutually adjacent vertices (*clique*) of G.

Throughout the paper the words *vertex*, *interval*, and *item* will be used interchangeably, and the subsets of a feasible solution to $BPPC$, BP, and VC will be called *colors* or *bins*.

3. Literature review

$BPPC$ is widely discussed in literature. We start by surveying some papers where approaches to $BPPC$ with arbitrary conflict graphs are proposed, then we discuss some issues about literature $BPPC$ instances and some papers devoted to $BPPC$ on interval graphs.

Kalfakakou et al. (2003) present a heuristic algorithm which repeatedly create a new subset with weight close to B from a maximal independent subset. Gendreau et al. (2004) propose a lower bound and six heuristics: one is a direct adaptation of the First-Fit Decreasing algorithm by Johnson (1974), three are based on graph coloring, and two are based on finding large cliques. They also describe a random arbitrary graph generator which, actually, outputs very peculiar graphs as discussed below (see also Section 6.1). Basnet and Wilson (2005) compare their heuristic algorithm with two of the best approaches by Gendreau et al. (2004), with the algorithm by Kalfakakou et al. (2003), and with a standard beam search algorithm: the proposed algorithm outperforms all the others, on average. Maiza and Guéret (2009) present a lower bound which improves those by Gendreau et al. (2004) and by Fernandes Muritiba et al. (2009) and is based on iterative runs of the lower bound algorithms by Gendreau et al. (2004). A Column Generation approach is proposed by Joncour et al. (2010) and tested on 280 instances with density between 10%-40% (the generation scheme is not specified). An exact algorithm based on a Set-Covering formulation is discussed by Fernandes Muritiba et al. (2010). The authors propose a very effective but time consuming lower bounds; several upper bounds are obtained by means of fast and good heuristic algorithms which are a parametrized adaptation of the classical First-Fit Decreasing, Best-Fit Decreasing, Worst-Fit Decreasing for BP Johnson (1974); lower and upper bounds are better than those by Gendreau et al. (2004). If no optimal solution is found then a population-based metaheuristic is applied and possibly a Branch-and-Price algorithm is adopted. Khanafer et al. (2010) improve some lower bounds by Fernandes Muritiba et al. (2010) by applying reduction procedures. Elhedhli et al. (2011) propose a Branch-and-Price algorithm which is compared with those by Fernandes Muritiba et al. (2010): results show that neither one outperforms the other. Maiza and Radjef (2011) propose seven heuristics: one is an adaptation of the Minimum Bin Slack heuristic by Gupta and Ho (1999); the others six repeatedly create a new subset by selecting (by means of classical Bin Packing methods) a subset of vertices from a maximal independent set previously generated. The experimental results show that these

heuristics outperform those by Gendreau et al. (2004). Yuan et al. (2014) show the effectiveness of an ant colony optimization approach to determine a feasible coloring solution to which an improved First-Fit Decreasing heuristic Bin Packing procedure is applied. Gschwind and Irnich (2016) describe an effective Column Generation approach to solve $BPPC$ and other problems to optimality, providing new classes of valid inequalities. Brandão and Pedroso (2016) present an exact method based on an arc-flow formulation with side constraints. The method builds very strong integer programming models that can be given in input to any state-of-the-art mixed integer programming solver. The algorithm is applied to many classical combinatorial problems and, in particular, all the instances by Fernandes Muritiba et al. (2010) are efficiently solved to optimality.

We remark that the generator by Gendreau et al. (2004) has been used to generate arbitrary graphs by many authors: Basnet and Wilson (2005), Brandão and Pedroso (2016), Capua et al. (2015), Clautiaux et al. (2011), Cornaz et al. (2017), Elhedhli et al. (2011), Gschwind and Irnich (2016), Joncour (2010), Joncour et al. (2010), Jouida et al. (2015), Khanafer (2010), Khanafer et al. (2012a), Khanafer et al. (2010), Khanafer et al. (2012b), Maiza and Guéret (2009), Maiza and Radjef (2011), Fernandes Muritiba (2010), Sadykov and Vanderbeck (2013), Yuan et al. (2014). In particular, Fernandes Muritiba et al. (2010) generated in this way the publicly available instances (see http://or.dei.unibo.it/library/bin-packing-problem-conflicts) which have been used by many of the authors above. In Section 6.1 we show that the graphs generated in this way are not arbitrary graphs but special interval graphs, namely threshold graphs. $BPPC$ on threshold graphs turns out to be easier than on arbitrary interval graphs and arbitrary graphs (see Bacci and Nicoloso (2017)).

Few papers are devoted to $BPPC$ with interval conflict graphs. Epstein and Levin (2008) present a $\frac{7}{3}$-approximation algorithm. Sadykov and Vanderbeck (2013) present a generic effective Branch-and-Price algorithm for the $BPPC$ with arbitrary conflict graphs, using a Dynamic Programming algorithm for pricing when the conflict graph is an interval graph. They test their algorithm on the instances by Fernandes Muritiba et al. (2010) and Elhedhli et al. (2011), closing all open instances, and on harder instances with an arbitrary conflict graph and a larger number of vertices per subset. A 2-approximation algorithm can be easily derived from the algorithm by Myung (2008) if G is a threshold graph. In fact, Myung (2008) studies the Minimum Clique Partitioning Problem on a weighted interval graph: given an interval graph with nonnegative vertex weights, find a partition of the vertices into the minimum number of cliques such that the sum of the vertex weights in each clique does not exceed a given bound B. Since G is a threshold graph, it is also a co-interval graph, hence $\overline{G}$, the complement of G, is an interval graph and $BPPC$ on a threshold graph G is equivalent to the Minimum Clique Partitioning Problem on the weighted interval graph $\overline{G}$. The special case of $BPPC$ when $w_i = 1$ for $i = 1, \ldots, n$ is the optimization version of Bounded Independent Sets (also known as Mutual Exclusion Scheduling, see Baker and Coffman (1996)), which is NP-complete when G is an interval graph and $B \geq 4$ (Bodlaender and Jansen (1995)). An application of Mutual Exclusion Scheduling on interval conflict graphs is described in Gardi (2009).

We recall the following problems related to $BPPC$. In Gupta et al. (2008) also item-bin conflicts are considered. When the number of subset is fixed, Kowalczyk and Leus (2016) minimize the weight of the heaviest subset, while Khanafer et al. (2012a) minimize the number of violated conflicts. Finally, given an instance of $BPPC$, let $D_p = \{i \in V : w_i = W_p\}$ be the set of items whose weight is equal to W_p, and let $d_p = |D_p|$. If we say that the items in D_p are *items of type* p and that d_p is their demand, then $BPPC$ is usually known as Cutting Stock problem (CS) (Delorme et al. (2016)) and it is formulated with integer variables and not binary ones like $BPPC$. CS arises in industrial contexts and often the number of different types of items is small w.r.t. the number n of items, while in $BPPC$ and BP it is not.

4. Interval Coloring algorithm with an Exchanging phase

Since $BPPC$ generalizes BP and VC, we believe that in order to design effective heuristic algorithms one has to adopt one of the following two approaches: modify an algorithm designed for BP or VC to directly obtain a feasible solution for $BPPC$, or determine a feasible solution for BP or VC, first, then modify it to obtain a feasible solution for $BPPC$.

For example, the first approach is used by the algorithms $U_{FF(\alpha)}$, $U_{BF(\alpha)}$, and $U_{WF(\alpha)}$ by Fernandes Muritiba et al. (2010): an algorithm designed for BP is modified in such a way that a vertex p is not assigned to a subset containing a vertex q in conflict with p. The first approach is applied also by Gendreau et al. (2004): an algorithm designed for VC is modified in such a way that when a new color S is created and its weight exceeds B, then a suitable subset $S' \subset S$ is determined such that $\sum_{i \in S \setminus S'} w_i \leq B$, and removed from S.

Here we adopt the second approach, as VC on interval graphs is solvable in linear time. In particular we construct an optimum feasible vertex coloring solution and, by a local search approach, we reduce the weight of the subsets whose weight exceeds B, if any, in order to obtain a feasible $BPPC$ solution.

Our Interval Coloring algorithm with an Exchanging phase (ICE) makes use of an interval model of G and of the following notations or definitions.

- $\lambda = \max\{LB_{BP}, \omega(G)\}$ is a lower bound for $BPPC$, where LB_{BP} denotes a lower bound for the Bin Packing problem underlying the given $BPPC$ and $\omega(G)$ denotes the size of a maximum clique of G (recall, in fact, that $\omega(G) = \chi(G)$ as G is an interval graph);
- $\mathcal{C}$ is the leftmost subset of $\omega(G)$ mutually intersecting intervals;
- $\pi = \max\{l_j, I_j \in \mathcal{C}\} + 0.5$ is a coordinate belonging to all the $\omega(G)$ intervals in $\mathcal{C}$;
- $\mathcal{I}_{left}$ is the set of intervals whose right endpoint lays on the left of π;
- $\mathcal{I}_{right}$ is the set of intervals whose left endpoint lays on the right of π;
- $W^{est}(V_i)$ is the *estimated weight* of the empty space on the right of the (unique) interval $I_j = (l_j, r_j)$ belonging to $V_i \cap \mathcal{C}$, if any, or on the right of π: assuming that the sum of all the interval weights is uniformly distributed among the $R \times \lambda$ unit segments, we define $W^{est}(V_i) = \mu(R - R_i)$ where $\mu = \frac{1}{R\lambda} \sum_{j=1,\ldots,n} w_j$ is the average weight of a unit segment, $R_i = r_j$ for $i = 1, \ldots, \omega(G)$, and $R_i = \pi$ for $i = \omega(G) + 1, \ldots, \lambda$;
- V_1, $\ldots$, V_z and z are the subsets in the current (possibly infeasible) solution and their number, respectively;
- $W(X)$ is the weight of a subset X of intervals, i.e. the sum of the weights of the intervals belonging to $X \subseteq V$; if $W(X) > B$ the subset X is *heavy*, otherwise it is *light*;
- $Tail(V_i, \rho) \subseteq V_i$, where $\rho \in [0, R]$ denotes a coordinate, is a subset of intervals of V_i which we define only for those i such that $\nexists I_j \in V_i : \; l_j < \rho < r_j$; when defined, $Tail(V_i, \rho) = \{I_h \in V_i : \; l_h \geq \rho\}$.
- a subset V_i is *non-conflicting* w.r.t. interval I_j if $V_i \cup \{I_j\}$ is an independent subset.

An example of the definitions above is illustrated in Figure 1 where a family $\mathcal{I}$ of $n = 12$ intervals is drawn. The pair $I_x(w_x)$ above each interval denotes the interval I_x and its weight w_x. In particular, $\mathcal{I} = \{I_1 = (4, 16)$, $I_2 = (12, 19)$, $I_3 = (17, 20)$, $I_4 = (2, 4)$, $I_5 = (11, 13)$, $I_6 = (17, 20)$, $I_7 = (6, 9)$, $I_8 = (15, 22)$, $I_9 = (0, 3)$, $I_{10} = (21, 24)$, $I_{11} = (5, 13)$, $I_{12} = (1, 10)\}$. The bound B for the considered instance of $BPPC$ is 100, and $R = 24$.

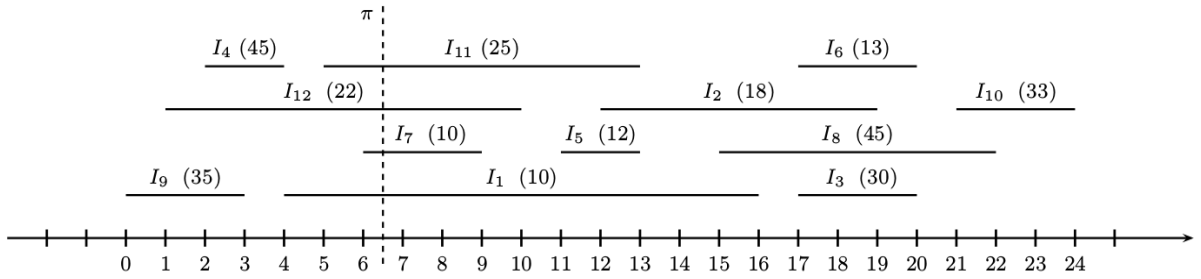

Figure 1: An example of a family of intervals $\mathcal{I}$: the pair $I_x(w_x)$ above each interval denotes the interval I_x and its weight w_x

Our algorithm makes use of the following data: $LB_{BP} = \left\lceil \sum_{i=1}^{12} w_i/B \right\rceil = \lceil \frac{298}{100} \rceil = 3$, $\omega(G) = |\mathcal{C}| = 4$ (as $\mathcal{C} = \{I_{11}, I_{12}, I_7, I_1\}$), hence $\lambda = \max\{LP_{BP}, \omega(G)\}$=4; $\pi = 6.5$; $\mathcal{I}_{left} = \{I_4, I_9\}$; $\mathcal{I}_{right} = \{I_6, I_2, I_{10}, I_5, I_8, I_3\}$; $R \times \lambda = 24 \times 4 = 96$; and $\mu = \frac{298}{96} = 3.1$.

Given the current solution V_1, V_2, V_3, V_4 where $V_1 = \{I_{11}\}$, $V_2 = \{I_{12}\}$, $V_3 = \{I_4, I_7\}$, $V_4 = \{I_1\}$, one has $R_1 = r_{11} = 13$, $R_2 = r_{12} = 10$, $R_3 = r_7 = 9$, $R_4 = r_1 = 16$, $W(V_1) = 25$, $W(V_2) = 22$, $W(V_3) = w_4 + w_7 = 55$, $W(V_4) = 10$, and all the subsets are light. By definition, $W^{est}(V_3) = \mu \times (R - R_i) = 3.1 \times (24 - 9) = 46.5$, $Tail(V_3, 5) = Tail(V_3, 6) = \{I_7\}$, $Tail(V_3, 9) = \emptyset$, and the subset V_3 is non-conflicting w.r.t. $I_5, I_2, I_6, I_8, I_3, I_{10}$.

The algorithm consists of two phases.

In the first phase the algorithm constructs a λ-coloring $\{V_1, \ldots, V_\lambda\}$ of G working on the chosen interval model of G. In particular, among all the feasible λ-colorings of G, the algorithm finds a coloring where the weight of the lightest subset is as large as possible, as we now describe.

It starts by assigning each interval of the leftmost maximum clique $\mathcal{C}$ to a different subset, then it assigns the intervals on the left of π, and finally those on the right of π. On the left of π, the algorithm repeatedly assigns an unassigned interval with rightmost right endpoint to the color V_i with minimum (current) weight $W(V_i) + W^{est}(V_i)$. On the right of π, the algorithm repeatedly assigns an unassigned interval with leftmost left endpoint to the color V_i with smallest (current) weight $W(V_i)$.

The algorithm ends the first phase with a feasible λ-coloring $\{V_1, \ldots, V_\lambda\}$ of G. If $W(V_i) \leq B$ for $i = 1, \ldots, \lambda$, the partition $\{V_1, \ldots, V_\lambda\}$ is a feasible solution for $BPPC$. Since its value λ equals the lower bound, it is also optimum for $BPPC$. If this is not the case, the algorithm proceeds with the second phase.

In the second phase the algorithm repeatedly selects a heaviest subset V_g and suitably modifies it to get a light subset. This is accomplished in two different ways: the TAIL-EXCHANGE, and the INSERTION.

Given a coordinate ρ, the TAIL-EXCHANGE between the chosen heavy subset V_g and a light subset V_h with minimum $W(Tail(V_h, \rho))$, consists of exchanging $Tail(V_g, \rho)$ with $Tail(V_h, \rho)$. It can be done if and only if the following three conditions are verified: both $Tail(V_g, \rho)$ and $Tail(V_h, \rho)$ are defined, $W(Tail(V_h, \rho)) < W(Tail(V_g, \rho))$, and the resulting V_h keeps being light. Notice that the weight of V_g after the exchange is decreased. Precisely, the algorithm finds the leftmost coordinate $\rho \geq \min\{r_s : I_s \in V_g\}$ such that there exists a subset V_h allowing a TAIL-EXCHANGE operation with V_g. If the resulting V_g is still heavy, the algorithm finds the next ρ with the same properties and repeats this step again, stopping as soon as $\rho \geq R$ or the current V_g is light. If V_g is still heavy the algorithm tries to apply the INSERTION.

In an INSERTION, an interval $I_j \in V_g$ which minimizes $|W(V_g) - w_j - B|$ is selected, and inserted into a light non-conflicting V_h such that the resulting V_h is light and $W(V_h) + w_j$ is maximum, if any. Otherwise I_j is inserted into a heavy non-conflicting V_h with minimum weight, if any.

The first time that no INSERTION is performed w.r.t. V_g, then a new subset is created and I_j is inserted into it.

The algorithm description follows. By TAIL-EXCHANGE STEP (INSERTION STEP, respectively) we mean the repeated application of a TAIL-EXCHANGE (INSERTION, respectively). Recall the one-to-one correspondence between intervals in $\mathcal{I}$ and vertices in V.

ALGORITHM ICE

Input: an interval model $\mathcal{I} = \{I_h = (l_h, r_h), h \in V\}$ for the graph $G = (V, E)$, $w_i \in \mathbb{Z}_+ \ \forall i \in V$, $B \in \mathbb{Z}_+$.
Output: z and a feasible partition $\{V_1, \ldots, V_z\}$ of V.

- PHASE I

  Define $z := \lambda$;
  Define z empty subsets V_1, ..., V_z;
  Assign each interval of $\mathcal{C}$ to a different subset;
  Let $I_j \in \mathcal{I}_{left}$ be an interval with rightmost right endpoint,
      assign I_j to a non-conflicting subset V_i with minimum $W(V_i) + W^{est}(V_i)$,
      remove I_j from $\mathcal{I}_{left}$, and repeat until $\mathcal{I}_{left}$ is empty;
  Let $I_j \in \mathcal{I}_{right}$ be an interval with leftmost left endpoint,
      assign I_j to a non-conflicting subset V_i with minimum $W(V_i)$,
      remove I_j from $\mathcal{I}_{right}$, and repeat until $\mathcal{I}_{right}$ is empty;

- PHASE II

  While $V_1, \ldots, V_z$ is infeasible do
      Let V_g be a subset with maximum weight;
      $\rho := \min\{r_s : I_s \in V_g\}$;
      TAIL-EXCHANGE STEP:
          While $W(V_g) > B$ and $\rho < R$ do
          let V_h, $h \neq g$, with minimum $W(Tail(V_h, \rho))$ be a light subset
          such that $W(Tail(V_h, \rho)) < W(Tail(V_g, \rho))$ and
          $W(V_h) - W(Tail(V_h, \rho)) + W(Tail(V_g, \rho)) \leq B$, if any;
          set $V_g := V_g \setminus Tail(V_g, \rho) \cup Tail(V_h, \rho)$
          and $V_h := V_h \setminus Tail(V_h, \rho) \cup Tail(V_g, \rho)$
          $\rho = \rho + 1$;
      new_subset:=FALSE;
      INSERTION STEP:
          While $W(V_g) > B$ do
          let $I_j \in V_g$ be an interval with minimum $|W(V_g) - B - w_j|$
          If there exists a V_h, $h \neq g$, non-conflicting w.r.t. I_j,
          such that $W(V_h) + w_j$ is maximum and $\leq B$,
          then remove I_j from V_g, and insert it in V_h,
          otherwise if there exists a V_h, $h \neq g$, non-conflicting
          w.r.t. I_j, such that $W(V_h)$ is minimum and $\geq B$,
          then remove I_j from V_g, and insert it in V_h,
          otherwise if new_subset=FALSE set $z := z + 1$ and new_subset:=TRUE
          then remove I_j from V_g and insert it in V_z.

If the partition at the end of Phase I is feasible then the algorithm terminates, otherwise the algorithm enters Phase II and chooses a heaviest subset V_g. If V_g after the TAIL-EXCHANGE STEP is light then the algorithm terminates an iteration of the main WHILE-instruction of Phase II and the number of heavy subsets is decreased by one, otherwise the algorithm performs the INSERTION STEP: if no new subset is created then, again, the algorithm terminates an iteration of the main WHILE-instruction of Phase II and the number of the heavy subsets is decreased by one; on the contrary, if a new subset V_{z+1} is created, the algorithm terminates this iteration of the main WHILE-instruction either with V_g and V_{z+1} light or with V_g light and V_{z+1} heavy but verifying $W(V_{z+1}) < W(V_g)$. In the former case the number of heavy subsets is decreased by one, in the latter the number of heavy subsets keeps constant but the overall infeasibility is decreased. This discussion shows that the algorithm terminates.

The computational complexity of Phase I is $O(n \log n)$. As for Phase II, the complexity of the TAIL-EXCHANGE STEP is $O(nR)$ and the complexity of the INSERTION STEP is $O(n^2)$. Since the main WHILE-instruction is repeated at most n times and $R \leq 2n$ (in fact, w.l.o.g., one can delete all the coordinates that do not host any endpoint), the overall computational complexity of Phase II is $O(n^3)$, and so is the complexity of the entire algorithm.

The algorithm has been tested over thousands of randomly generated instances. Computational results are presented in Section 6.

5. A random interval graph generator

The easiest way to generate a random interval graph is to randomly choose the endpoints of each of the n intervals and then construct the intersection graph of them (Vasileios (2005); Justicz et al. (1990)). Another (equivalent) generator is the following. Given a non-negative integer n, let $\sigma = (\sigma_1, \ldots, \sigma_{2n})$ be a random permutation of $(1, 1, 2, 2, \ldots, n, n)$. Then, for $j = 1, \ldots, n$ define interval $I_j = (l_j, r_j)$, where $l_j = \min\{k : \sigma_k = j, k = 1, \ldots, 2n\}$ and $r_j = \max\{k : \sigma_k = j, k = 1, \ldots, 2n\}$ (in fact, in σ there exist exactly two elements of value j). We generated thousands of sets of intervals in both ways. The experimental analysis we conducted shows that the edge density $2|E|/(n(n-1))$ of almost all the corresponding interval graphs is 60%-70%. The *BPPC* instances in the literature (see, for example, Gendreau et al. (2004); Fernandes Muritiba et al. (2010); Sadykov and Vanderbeck (2013)) are classified by their edge density ranging from 0% to 90%. Since, to our knowledge, no random interval graph generator exists which allows to obtain an interval graph with a prescribed edge density, in the present section we define a new random interval graph generator with this property. In fact, the edge density of the intersection graph of a set of intervals in $[0; D]$ depends on the average interval length.

Our generator accepts in input the number n of intervals and the expected edge density Δ of the corresponding intersection graph, and suitably computes three values, D, $\Lambda_{\min}$, $\Lambda_{\max}$ to obtain an interval graph with expected edge density Δ. The output consists of a set of n intervals whose endpoints (integer, w.l.o.g.) are uniformly distributed in $\{0, \ldots, D\}$, and an arbitrary interval $I_j = (l_j, r_j)$ in this set will have length $\Lambda_j = r_j - l_j$ verifying $\Lambda_{\min} \leq \Lambda_j \leq \Lambda_{\max}$. Experimental analysis conducted on thousands of graphs confirms that our generator generates arbitrary interval graph with average edge density Δ. Precisely, the edge density δ of an interval graph generated in this way almost always verifies $\delta \in [\Delta - \epsilon(n); \Delta + \epsilon(n)]$, where $\epsilon(n)$ decreases for increasing n (in particular, $\epsilon(120) = 0.05$ and $\epsilon(1000) = 0.025$). The details follow.

In order to ensure that the $2n$ endpoints of the n intervals have space enough to give a graph with edge density equal to zero (for suitable interval lengths), one has to fix $D \geq 2n$. We tried many different values for D but we did not appreciate any differences, so we decided to set $D = 2.5n$.

The edge density Δ of the resulting interval graph depends on the average interval length $\overline{\Lambda}$: for example when $\overline{\Lambda} = 1$ then $\Delta \simeq 0$, and when $\overline{\Lambda} > D/2$ then $\Delta \simeq 1$. We can determine

the equation which ties Δ and $\overline{\Lambda}$ reasoning as follows. The coordinate 0 can be chosen as a left endpoint, only, the coordinate D can be chosen as a right endpoint, only, and all the other coordinates can be chosen both as left and right endpoints. Hence the average number of left endpoints per coordinate is $\frac{n}{D}$, as well as the average number of right endpoints per coordinate, and an interval of length Λ intersects $\frac{2n}{D}\Lambda$ intervals, on average. Thus the average degree of a vertex is $\frac{2n}{D}\overline{\Lambda}$, resulting in $\frac{2n}{D}\frac{n}{2}\overline{\Lambda}$ edges of the interval graph. Dividing this quantity by $\frac{n(n-1)}{2}$, one gets that the edge density is $\Delta = \frac{2n}{(n-1)D}\overline{\Lambda}$ from which we derive that the average interval length $\overline{\Lambda}$ has to be set to $\Delta D\frac{n-1}{2n}$ in order to obtain an interval graph with expected edge density Δ.

Recalling that $\overline{\Lambda} = 1/n\sum_{j=1}^{n}\Lambda_j$ and that $\Lambda_{\min} \leq \Lambda_j \leq \Lambda_{\max}$ for all j, the set of intervals has average length $\overline{\Lambda}$ if one suitably chooses $\Lambda_{\min}$ and $\Lambda_{\max}$. We decided to randomly choose $\Lambda_{\min}$ in a suitable range which we discuss in a while; $\Lambda_{\max}$ is consequently determined. We reason as follows.

By flipping a coin, the generator randomly chooses the left endpoint l_j of interval I_j first, or the right endpoint r_j. Assume that the left endpoint l_j is randomly chosen first. In order to ensure that the length Λ_j verifies $\Lambda_j \geq \Lambda_{\min}$, l_j has to be chosen in $\{0,\ldots,D-\Lambda_{\min}\}$. Refer to Figure 2. If $l_j \in \{0,\ldots,D-\Lambda_{\max}\}$ then r_j can be randomly chosen in $\{l_j+\Lambda_{\min},\ldots,l_j+\Lambda_{\max}\}$. In this case the expected interval length $\mathrm{EIL}(l_j)$ is $(\Lambda_{\min}+\Lambda_{\max})/2$. If $l_j \in \{D-\Lambda_{\max}+1,\ldots,D-\Lambda_{\min}\}$ then r_j can be randomly chosen only in $\{l_j+\Lambda_{\min},\ldots,D\}$ and $\mathrm{EIL}(l_j)$ is a linear function of l_j, namely $(D+\Lambda_{\min}-l_j)/2$. Thus the average interval length $\overline{\Lambda}'$ when l_j is chosen first can be computed dividing the entire area underlying the drawn function by its width $D-\Lambda_{\min}$ as follows:

$$\overline{\Lambda}' = \frac{1}{D-\Lambda_{\min}}\big[(D-\Lambda_{\max})\frac{\Lambda_{\min}+\Lambda_{\max}}{2} + (\frac{\Lambda_{\max}-\Lambda_{\min}}{2})(\frac{\Lambda_{\min}+\Lambda_{\max}}{2}+\Lambda_{\min})\big] =$$

$$= \frac{1}{4(D-\Lambda_{\min})}[-\Lambda_{\max}^2 - 3\Lambda_{\min}^2 + 2D(\Lambda_{\max}+\Lambda_{\min})]$$

Assume now that the right endpoint r_j of interval I_j is randomly chosen first. By similar arguments, r_j has to be chosen in $\{\Lambda_{\min},\ldots,D\}$ and $l_j \in \{\max\{r_j-\Lambda_{\max};0\},\ldots,r_j-\Lambda_{\min}\}$ and the average interval length $\overline{\Lambda}''$ when r_j is chosen first is equal to $\overline{\Lambda}'$ (the graph of $\mathrm{EIL}(r_j)$ can be obtained by horizontally flipping the graph of Figure 2 along the vertical axis $D/2$).

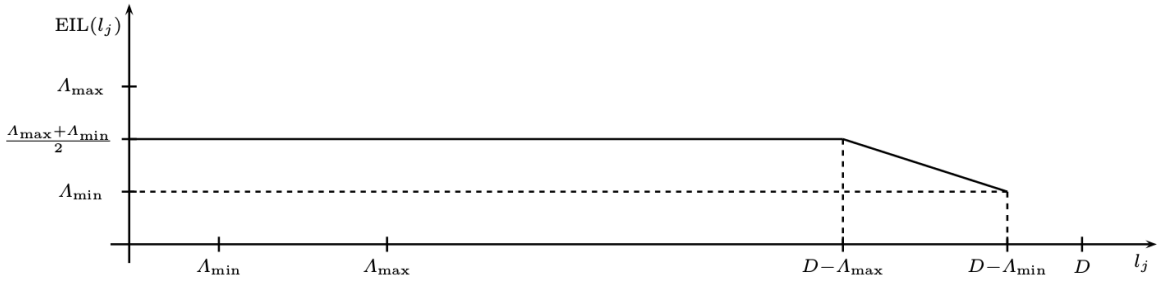

Figure 2: The expected interval length $\mathrm{EIL}(l_j)$ when l_j is randomly chosen

Since, on average, one half of the intervals are generated by randomly choosing the left endpoint first, and one half by randomly choosing the right endpoint first, the overall average interval length is

$$\overline{\Lambda} = \frac{\overline{\Lambda}'+\overline{\Lambda}''}{2} = \frac{1}{4(D-\Lambda_{\min})}[-\Lambda_{\max}^2 - 3\Lambda_{\min}^2 + 2D(\Lambda_{\max}+\Lambda_{\min})]$$

Given $\Lambda_{\min}$ and $\overline{\Lambda}$, we can use this formula to determine $\Lambda_{\max}$. Since $\Lambda_{\max} > D$ makes no sense, we get

$$\Lambda_{\max} = D - \sqrt{D^2 - 4D\overline{\Lambda} + 2D\Lambda_{\min} + 4\overline{\Lambda}\Lambda_{\min} - 3\Lambda_{\min}^2}$$

The non-negativity of the argument of the square root requires $\frac{4\,\overline{\Lambda}-D}{3} \leq \Lambda_{\min} \leq D$. On the other hand, clearly, $1 \leq \Lambda_{\min} \leq \overline{\Lambda}$. Hence $\max\{1; \lceil \frac{4\,\overline{\Lambda}-D}{3} \rceil\} \leq \Lambda_{\min} \leq \overline{\Lambda}$, as $\overline{\Lambda} \leq D$.

The proposed generator fixes D and computes $\overline{\Lambda}$ as discussed. Then, for each interval I_j, it randomly chooses $\Lambda_{\min} \in \left\{\max\{1; \lceil \frac{4\,\overline{\Lambda}-D}{3} \rceil\}, \ldots, \overline{\Lambda}\right\}$, computes the corresponding $\Lambda_{\max}$, randomly chooses whether to generate r_j first or l_j and, in both cases, randomly generates a suitable Λ_j, and consequently fixes the other endpoint.

RANDOM INTERVAL GRAPH GENERATOR

Input: $n \in \mathbb{Z}_+$ and $\Delta \in [0,1]$
Output: a set $\mathcal{I} = \{I_h = (l_h, r_h), h = 1...n\}$ of intervals whose intersection graph has expected edge density Δ.

$D := 2.5\, n$;
$\overline{\Lambda} := \Delta \cdot D^{\frac{n-1}{2n}}$;
For $j = 1, \ldots, n$
  randomly choose $\Lambda_{\min} \in \{ \max\{1; \lceil \frac{4\,\overline{\Lambda}-D}{3} \rceil\}, \ldots, \overline{\Lambda} \}$
  $\Lambda_{\max} := D - \sqrt{D^2 - 4D\overline{\Lambda} + 2D\Lambda_{\min} + 4\overline{\Lambda}\Lambda_{\min} - 3\Lambda_{\min}^2}$;
  flip a *coin*;
  if *coin* = HEAD then randomly choose $r_j \in \{\Lambda_{\min}, \ldots, D\}$ and
    $\Lambda_j \in \{\Lambda_{\min}, \ldots, \min\{r_j; \Lambda_{\max}\}\}$, and set $l_j := r_j - \Lambda_j$;
  if *coin* = TAIL then randomly choose $l_j \in \{0, \ldots, D - \Lambda_{\min}\}$ and
    $\Lambda_j \in \{\Lambda_{\min}, \ldots, \min\{D - l_j; \Lambda_{\max}\}\}$, and set $r_j := l_j + \Lambda_j$.

6. Computational results

In the present section we compare the results obtained by running our algorithm and some of the most performing heuristic procedures taken from the literature, which we now describe.

To our knowledge, no heuristic algorithms exist specifically designed for solving $BPPC$ on interval graphs. However, some heuristic algorithms are presented in the literature for solving $BPPC$ on arbitrary graphs: precisely, Kalfakakou et al. (2003), Gendreau et al. (2004), Basnet and Wilson (2005), Fernandes Muritiba et al. (2010), Maiza and Radjef (2011), and Yuan et al. (2014).

The heuristic by Kalfakakou et al. (2003) searches for a maximal independent subset with total weight as large as possible and less than or equal to B, its vertices are removed from the vertex set, and the process is repeated until all the vertices are considered.

Gendreau et al. (2004) define six heuristic procedures, $H1$ to $H6$. According to their computational results, $H6$ is usually the best one and it works as follows. Let H be the set of vertices with degree zero in the extended conflict graph G_w (obtained by adding to G an edge for each pair of vertices i, j with $w_i + w_j > B$) and $V' := V \setminus H$. The algorithm starts by computing a maximal clique D of the graph $G_w(V')$ induced by V' and then sets $V' := V' \setminus D$. Then, for each vertex $i \in D$ it computes a maximal independent set D_i on the graph $G_w(\{i\} \cup V')$. It solves a classical BP on D_i using the First-Fit Decreasing algorithm: only the subset S_i containing i becomes part of the final solution. It updates $V' := V' \setminus S_i$ and $D := D \setminus \{i\}$. If $V' = \emptyset$, it

assigns the vertices in H to the existing subsets and possibly to new ones using the First-Fit Decreasing algorithm. If $V' \neq \emptyset$ and $D = \emptyset$, it repeats by computing a new maximal clique D of the current $G_w(V')$. If $V' \neq \emptyset$ and $D \neq \emptyset$, it selects a new vertex $i \in D$ and starts by computing a maximal independent set again.

Two new heuristic procedures called MGH and BS are described in Basnet and Wilson (2005). MGH is an adaptation of the classical Best Fit Decreasing heuristic (Johnson (1974)) designed for BP with items sorted by non-increasing degree in G breaking ties by non-increasing weights. BS is based on a standard beam search algorithm: at each level of the enumeration tree produced by the BFS by Christofides (1975), a limited number of vertices is sprouted according to the highest space utilization.

Fernandes Muritiba et al. (2010) first describe three parametrized heuristic algorithms that are an adaptation to $BPPC$ of the classical First-Fit Decreasing, Best-Fit Decreasing, Worst-Fit Decreasing for BP (Johnson (1974)). In particular, these adaptations, $U_{FF(\alpha)}$, $U_{BF(\alpha)}$, and $U_{WF(\alpha)}$ (we shall call them algorithms M), consider the extended conflict graph G_w and modified vertex weights w_i^s defined as follows: $w_i^s = \alpha(w_i/\overline{w}) + (1-\alpha)(\deg(i)/\overline{\deg})$, for $i = 1, 2, \ldots, n$, where $\alpha \in \{0, 0.1, \ldots, 1\}$, $\deg(i)$ is the degree of vertex i in G_w, and $\overline{w}$ and $\overline{\deg}$ are the average weight of the vertices and their average degree in G_w, respectively. If the best of the 33 solution values found by these heuristic algorithms is not equal to the lower bound proposed by the authors, then the authors invoke a metaheuristic procedure, the Population Heuristic (PH), consisting in a Tabu Search enriched with a diversification step based on a crossover operator.

Maiza and Radjef (2011) describe seven heuristic algorithms: according to their computational results, the best one is $H_{W_1}^{MBS}$ on average. The procedure works as $H6$ except for the algorithm used for solving the classical BP on D_i: in this case the MBS algorithm by Gupta and Ho (1999) is used, which repeatedly finds a set of vertices whose weight is as large as possible without exceeding B.

Yuan et al. (2014) propose an Ant Colony Optimization (ACO) based heuristic algorithm: a feasible vertex coloring on G is found by the ACO procedure and then a First Fit Decreasing algorithm is applied to the vertices in the same subset.

Both algorithms M and $H_{W_1}^{MBS}$ perform better than $H6$, as discussed in Maiza and Radjef (2011) and in Fernandes Muritiba et al. (2010). On the other hand, the authors of the ACO algorithm declare that "*with the increasing number of items, the limitation of the algorithm is exposed*". In fact, their experiments are conducted on instances with no more than 500 vertices, while we run our algorithm for instances with 1000 vertices. For these reasons, in the present section, we compare our heuristic algorithm with the algorithms M, MGH, and $H_{W_1}^{MBS}$. We also decided not to implement PH by Fernandes Muritiba et al. (2010) because its running time largely exceeds the computing time of our algorithm. In fact, the authors themselves set a time limit of 120 seconds. We remark that the solution output by our algorithm could be easily included in the initial pool of solutions needed by PH.

We coded algorithms M, MGH, $H_{W_1}^{MBS}$ and our heuristic in C++, and run them on an Intel Core i7-3632QM 2.20GHz with 16 GB RAM under a Linux operating system. In Section 6.1 we discuss the computational results on instances taken from the literature. Since these instances are characterized by very peculiar interval conflict graphs (see discussion below), we also discuss the computational results on instances with arbitrary interval conflict graphs (see Section 6.2).

6.1. Literature instances

In this section we discuss the results obtained by running algorithms M, MGH, $H_{W_1}^{MBS}$, and ICE on the instances by Fernandes Muritiba et al. (2010) (see http://www.or.deis.unibo.it) and on other instances generated as described in Sadykov and Vanderbeck (2013). The conflict graphs of the instances by Fernandes Muritiba et al. (2010) have been generated by means of the generator by Gendreau et al. (2004) with the intention of obtaining arbitrary graphs. Gendreau

et al. (2004) describe the generator, which we shall refer to as T-*generator*, as follows: "*A value* p_i *was first assigned to each vertex* $i \in V$ *according to a continuous uniform distribution on* $[0, 1]$. *Each edge* (i, j) *of* G *was created whenever* $(p_i + p_j)/2 \leq d$, *where* d *is the expected density of* G.". Sadykov and Vanderbeck (2013) observed that this generation scheme results in interval graphs. Actually, we noticed that it results in very peculiar interval graphs, precisely threshold ones (defined in a while, see also Figure 3), and that the expected edge density is a function $f(d)$ of d, and not d as claimed (Bacci and Nicoloso (2017)). In Table 1, the value d and the corresponding $f(d)$ are shown.

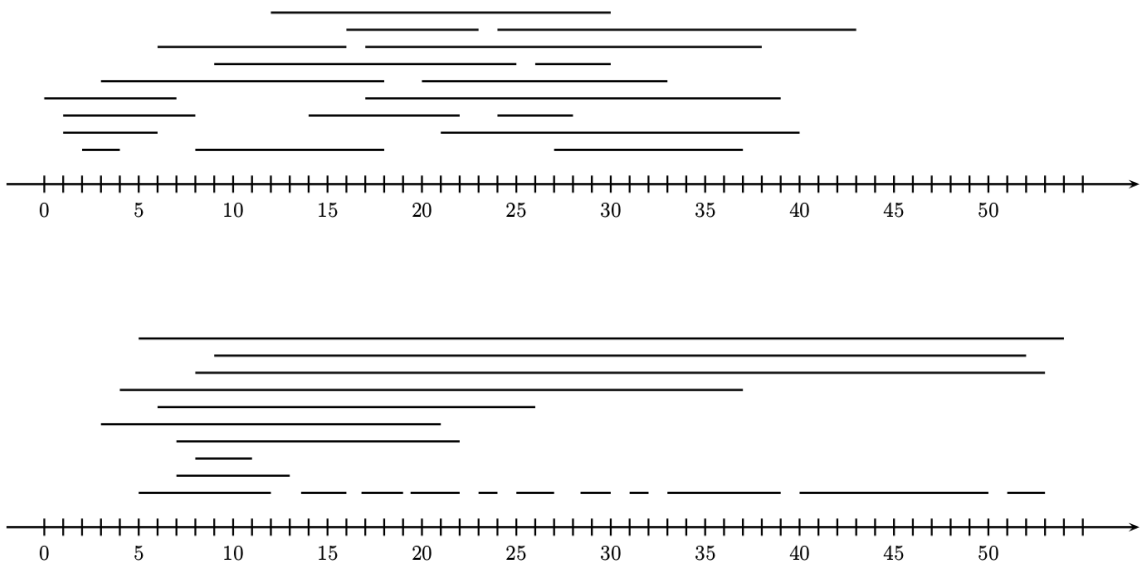

Figure 3: An interval representation of an interval graph generated with the generator of Section 5 (above) and of a threshold graph generated by means of the generator by Gendreau et al. (2004) (below). Both graphs have 20 vertices and an edge density equal to 50%.

A graph is a threshold graph if there exist a real number d (the threshold) and a weight p_x for every vertex x such that (i, j) is an edge iff $(p_i + p_j)/2 \leq d$ (Chvátal and Hammer (1973)). A threshold graph is at the same time an interval graph, a co-interval graph, a cograph, a split graph, and a permutation graph (Golumbic (1980)). In addition, its complement, where (i, j) is an edge iff $(p_i + p_j)/2 > d$, is a threshold graph too. VC is solvable in linear time on threshold graphs, nevertheless $BPPC$ with a threshold conflict graph remains NP-hard

The T-generator clearly produces threshold graphs. This generator has been used by many authors (see list in Section 3), in particular by Fernandes Muritiba et al. (2010) for generating the publicly available instances.

Threshold graphs have a very particular structure that allows, unlike an arbitrary interval graph, to partition the vertices into a unique maximum clique and a stable set. To appreciate the difference that exists between an interval graph and a threshold one, in Figure 3 we draw a set of intervals generated with the generator of Section 5 (above) and an interval model of graph obtained with the generator by Gendreau et al. (2004) (below): the edge density of the intersection graphs of both of them is equal to 0.5.

By $TM(n, B, f(d))$ we denote a set of ten instances with n vertices, bound B, and threshold conflict graph with density $f(d)$, where $n \in \{120, 250, 500, 1000\}$, $B \in \{120, 150, \ldots, 390\} \cup \{400\}$, and $d \in \{0, 0.1, \ldots, 0.9\}$. In particular, the weights and the conflict graphs of all the $TM(n, B, f(d))$ are exactly those in the classes 1,2,3,4 by Fernandes Muritiba et al. (2010). Their instances were built selecting the first 10 instances of the 20 originally proposed by Falkenauer (1996) for

the Bin Packing (without conflicts), and adding 10 random threshold conflict graphs generated by means of the T-generator, varying d from 0 to 0.9. The Bin Packing instances proposed by Falkenauer (1996) have weights uniformly distributed in $[20, 100]$ and $B = 150$ because, as the author says, this setup was the most difficult for the Bin Packing lower bound algorithms by Martello and Toth (1990). Nevertheless, Gent (1998) easily solves the last five open instances. We remark that most of the authors in the literature consider $B = 150$, only.

In order to verify how much the weights affect the quality of the solution and/or the computing time, we also decided to construct the TS instances: by $TS(n, B, f(d))$ we will denote a set of ten instances with n vertices, bound B, and threshold conflict graph with density $f(d)$, where n and d are defined as above, while $B \in \{3000, 3750, \ldots, 9750\} \cup \{10000\}$. The conflict graphs of a $TS(n, \cdot, f(d))$ are those of $TM(n, \cdot, f(d))$. As for the weights, they are uniformly distributed in $[500, 2500]$, as the "*instances with a larger number of items per bin*" by Sadykov and Vanderbeck (2013) (the so-called "d instances"). We remark that the only TS instances solved by Sadykov and Vanderbeck (2013) have $B = 10000$.

Let $\overline{w}$ be the average weight of a vertex, then the average number of items per subset (ANIS for short) is $B/\overline{w}$. It is worth observing that the same ANIS is obtained in the $TM(n, B, f(d))$ and in the $TS(n, 25 \times B, f(d))$ instances. Observe also that in the TS instances the number of different weights is $2500 - 500 + 1 = 2001$. Hence, every weight is expected to appear in $n/2001$ copies. For a same n, the (classical) Bin Packing underlying a TM instance recalls a Cutting Stock problem (see Section 3) in a stronger way than the one underlying a TS instance.

The detailed results obtained for $n = 1000$ are presented in Table 1, where rows are indexed by d and $f(d)$ and columns by ANIS and B; the results for $n \in \{120, 250, 500, 1000\}$ are summarized in Table 2.

Let X be an instance of $BPPC$ with conflict graph G. Then, $MS(X)$ denotes the minimum solution value among those output by MGH, $H^{MBS}_{W_1}$, and algorithms M on the instance X and ICE(X) denotes the value of the solution output by algorithm ICE on X. To evaluate the performances of the algorithms we define $LB_{BPPC}(X) = \max\left\{\left\lceil \sum_{i \in V} w_i/B \right\rceil; \chi(G)\right\}$, a lower bound on the value of an optimum solution of $BPPC$ on instance X.

In each cell of the tables there are six values, each one averaged over the cardinality of the corresponding set of instances: MS=LB (ICE=LB, respectively) is the percentage of instances X where $MS(X)$=$LB_{BPPC}(X)$ (ICE(X)=$LB_{BPPC}(X)$, respectively), i.e. the percentage of instances where $LB_{BPPC}(X)$ allows to certify that the corresponding algorithm found an optimum solution; MS<ICE (ICE<MS, respectively) is the percentage of instances X where $MS(X) <$ ICE(X) (ICE$(X) < MS(X)$, respectively) (notice that the complement to 100% of the sum of the last two values is the percentage of instances where $MS(X)$=ICE(X)); Gap_MS (Gap_ICE, respectively) is the gap $\frac{MS(X) - LB_{BPPC}(X)}{LB_{BPPC}(X)}$ ($\frac{\text{ICE}(X) - LB_{BPPC}(X)}{LB_{BPPC}(X)}$, respectively). The best data in a cell are highlighted in grey. If in a cell the value MS=LB (ICE=LB) is 100% then the rows containing the data of the algorithms M, MGH, $H^{MBS}_{W_1}$ (the algorithm ICE) are colored in cyan.

The computational results give evidence that the best solution among those output by algorithms M, MGH, $H^{MBS}_{W_1}$ is better than ours when ANIS ≤ 2.5. We believe that this is mainly due to the fact that algorithms M, MGH, $H^{MBS}_{W_1}$ take advantage of the informations in the extended conflict graph G_w. Indeed, when ANIS ≤ 2.5, many are the edges added to G to obtain G_w, resulting in a G_w which is not a threshold graph anymore.

When ANIS $= 3$ or when ANIS ≥ 3.5 and $d \leq 0.2$ our algorithm is the best approach. Notice the very good performance of algorithm ICE for $d = 0$, that is to say on instances of Bin Packing (without conflicts) with three or more vertices per subset. We believe that our algorithm takes advantage of the local search approach in the second phase, which rearranges the solution found in PHASE I.

When ANIS ≥ 3.5 and $d \geq 0.3$, the value of the best solution among those output by algorithms M, MGH, $H^{MBS}_{W_1}$ is equal to ours for most of the instances. Probably this is due to the following facts:

			ANIS	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	6.7
			B	120	150	180	210	240	270	300	330	360	390	400
d	$f(d)$		MS=LB	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
			MS<ICE	100%	50%	0%	0%	0%	0%	0%	0%	0%	0%	0%
		0.0	Gap_MS	1.98%	1.22%	0.84%	1.46%	0.84%	0.94%	0.99%	0.55%	0.77%	0.65%	0.86%
		0.0	ICE=LB	0%	0%	10%	10%	70%	80%	100%	100%	90%	100%	80%
			ICE<MS	0%	20%	100%	100%	100%	100%	100%	100%	100%	100%	90%
			Gap_ICE	7.70%	1.37%	0.45%	0.31%	0.12%	0.09%	0.00%	0.00%	0.06%	0.00%	0.13%
			MS=LB	0%	0%	0%	0%	0%	0%	0%	0%	30%	20%	10%
			MS<ICE	100%	20%	0%	0%	0%	0%	0%	0%	0%	0%	0%
		0.1	Gap_MS	1.98%	1.15%	0.84%	0.94%	0.84%	0.81%	0.50%	0.60%	0.48%	0.52%	0.60%
		0.02	ICE=LB	0%	0%	10%	10%	70%	80%	100%	100%	70%	100%	90%
			ICE<MS	0%	40%	100%	100%	100%	100%	100%	100%	50%	80%	80%
			Gap_ICE	7.92%	1.17%	0.48%	0.31%	0.12%	0.09%	0.00%	0.00%	0.18%	0.00%	0.07%
			MS=LB	0%	0%	0%	0%	0%	0%	60%	90%	100%	100%	100%
			MS<ICE	100%	50%	0%	0%	0%	10%	0%	0%	0%	0%	0%
		0.2	Gap_MS	1.98%	1.30%	1.28%	1.08%	0.87%	0.89%	0.35%	0.05%	0.00%	0.00%	0.00%
		0.08	ICE=LB	0%	0%	0%	0%	30%	60%	100%	100%	100%	100%	100%
			ICE<MS	0%	0%	100%	100%	80%	90%	40%	10%	0%	0%	0%
			Gap_ICE	8.26%	1.59%	0.60%	0.42%	0.28%	0.22%	0.00%	0.00%	0.00%	0.00%	0.00%
			MS=LB	0%	0%	0%	70%	80%	90%	90%	100%	100%	100%	100%
			MS<ICE	100%	90%	10%	10%	10%	0%	0%	0%	0%	0%	0%
		0.3	Gap_MS	1.98%	1.42%	1.70%	0.21%	0.07%	0.03%	0.03%	0.00%	0.00%	0.00%	0.00%
		0.18	ICE=LB	0%	0%	0%	60%	70%	90%	90%	100%	100%	100%	100%
			ICE<MS	0%	0%	60%	10%	0%	0%	0%	0%	0%	0%	0%
			Gap_ICE	7.84%	2.54%	1.43%	0.20%	0.10%	0.03%	0.03%	0.00%	0.00%	0.00%	0.00%
			MS=LB	0%	0%	10%	30%	60%	60%	80%	80%	90%	90%	90%
			MS<ICE	100%	100%	0%	10%	0%	0%	0%	0%	0%	0%	0%
		0.4	Gap_MS	2.14%	1.84%	0.57%	0.20%	0.12%	0.10%	0.05%	0.05%	0.02%	0.02%	0.02%
		0.32	ICE=LB	0%	0%	20%	30%	60%	60%	80%	80%	90%	90%	90%
			ICE<MS	0%	0%	70%	0%	0%	0%	0%	0%	0%	0%	0%
			Gap_ICE	9.23%	4.84%	0.32%	0.22%	0.12%	0.10%	0.05%	0.05%	0.02%	0.02%	0.02%
			MS=LB	0%	0%	30%	90%	100%	100%	100%	100%	100%	100%	100%
			MS<ICE	100%	90%	0%	0%	0%	0%	0%	0%	0%	0%	0%
		0.5	Gap_MS	7.37%	1.11%	0.22%	0.02%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		0.5	ICE=LB	0%	0%	60%	90%	100%	100%	100%	100%	100%	100%	100%
			ICE<MS	0%	0%	40%	0%	0%	0%	0%	0%	0%	0%	0%
			Gap_ICE	14.88%	2.01%	0.12%	0.02%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
			MS=LB	0%	0%	20%	80%	80%	80%	80%	80%	80%	90%	90%
			MS<ICE	100%	80%	20%	0%	0%	0%	0%	0%	0%	0%	0%
		0.6	Gap_MS	5.97%	0.73%	0.20%	0.05%	0.03%	0.03%	0.03%	0.03%	0.03%	0.02%	0.02%
		0.68	ICE=LB	0%	0%	40%	80%	80%	80%	80%	80%	80%	90%	90%
			ICE<MS	0%	0%	40%	0%	0%	0%	0%	0%	0%	0%	0%
			Gap_ICE	11.00%	1.53%	0.17%	0.05%	0.03%	0.03%	0.03%	0.03%	0.03%	0.02%	0.02%
			MS=LB	0%	0%	50%	80%	90%	90%	90%	100%	100%	100%	100%
			MS<ICE	100%	80%	10%	10%	0%	10%	0%	0%	0%	0%	0%
		0.7	Gap_MS	4.24%	0.73%	0.14%	0.06%	0.03%	0.01%	0.01%	0.00%	0.00%	0.00%	0.00%
		0.82	ICE=LB	0%	0%	60%	80%	90%	90%	90%	100%	100%	100%	100%
			ICE<MS	0%	0%	10%	0%	0%	0%	0%	0%	0%	0%	0%
			Gap_ICE	7.58%	1.26%	0.14%	0.07%	0.03%	0.03%	0.01%	0.00%	0.00%	0.00%	0.00%
			MS=LB	0%	0%	30%	70%	70%	90%	90%	90%	90%	100%	100%
			MS<ICE	100%	100%	20%	20%	20%	0%	0%	0%	0%	0%	0%
		0.8	Gap_MS	2.96%	0.48%	0.15%	0.06%	0.05%	0.01%	0.01%	0.01%	0.01%	0.00%	0.00%
		0.92	ICE=LB	0%	0%	30%	60%	60%	90%	90%	90%	90%	100%	100%
			ICE<MS	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
			Gap_ICE	4.73%	0.97%	0.19%	0.09%	0.07%	0.01%	0.01%	0.01%	0.01%	0.00%	0.00%
			MS=LB	0%	10%	60%	100%	100%	100%	100%	100%	100%	100%	100%
			MS<ICE	100%	80%	0%	0%	0%	0%	0%	0%	0%	0%	0%
		0.9	Gap_MS	1.61%	0.23%	0.04%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		0.98	ICE=LB	0%	0%	60%	100%	100%	100%	100%	100%	100%	100%	100%
			ICE<MS	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
			Gap_ICE	2.25%	0.55%	0.04%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
			tmin_ICE	0.0520	0.0399	0.0345	0.0294	0.0283	0.0262	0.0243	0.0219	0.0217	0.0215	0.0200
			tmax_ICE	2.6765	0.9633	0.5018	0.3485	0.1616	0.1247	0.0917	0.0739	0.0748	0.0637	0.0698
			tavg_ICE	0.9639	0.3169	0.1593	0.1041	0.0647	0.0526	0.0430	0.0375	0.0358	0.0334	0.0356

Table 1: Computational results on $TM(1000, B, \Delta)$. In grey, the best data of a cell; In cyan, the data of the algorithm with 100% of certified optimum.

- the extended conflict graph G_w is a threshold graph because it coincides with G; hence, it contains a unique maximum clique $\omega(G)$ of size $n \times d$ (Bacci and Nicoloso (2017)) and the remaining vertices form a stable set;
- by definition, $LB_{BP} = \lceil \frac{n \times \overline{w}}{B} \rceil = \lceil \frac{n}{\text{ANIS}} \rceil$.
- for the considered values of ANIS and d, we have that $\omega(G) > LB_{BP}$, hence the assignment of the mutually non-conflicting vertices of the stable set is mainly due to the conflicts in G and not to the weights

From what above, since the maximum clique of G is unique, all the algorithms output similar solutions.

The cyan cells in Table 1 show that algorithm ICE solves to optimality all the 10 instances of 36 out of 100 cells, while the the best solution among those output by algorithms M, MGH, $H_{W_1}^{MBS}$ in 28 out of 100 cells. Analyzing the computational experiments into details, we also noticed that, among MGH, $H_{W_1}^{MBS}$, and M, algorithm MGH rarely is the best, and that $H_{W_1}^{MBS}$ in most of the cases outperforms the others when the edge density of the conflict graph is zero.

The minimum, maximum, and average time (tmin_ICE, tmax_ICE, tavg_ICE, respectively) in seconds required by ICE to solve one instance out of the 10 in each cell are reported at the bottom of each column of Table 1. Basically these times decrease for increasing ANIS: we think that this is due to the reduced number of operations in PHASE II. We also measured the average times required by the other algorithms. Precisely, M and MGH always require less than 0.1 seconds to solve one instance out of the 10 in each cell for all B and d, while $H_{W_1}^{MBS}$ requires 0.8 seconds, its maximum being 5.8 seconds on one instance when $B = 150$ and $d = 0.2$.

More general results are shown in Table 2, where we compare the data output by the heuristic algorithms M, MGH, $H_{W_1}^{MBS}$, and ICE over $TM(n, B, f(d))$, with $n \in \{120, 250, 500, 1000\}$ (in the first column the results are averaged over 2000 instances, in second column over 1000, in the column third over 8000, and in the last column over 11000, and in all cases averaged overall values of d).

The results of the heuristic algorithms M, MGH, $H_{W_1}^{MBS}$, and ICE over $TS(n, B, f(d))$, with $n \in \{120, 250, 500, 1000\}$, can be found in Table 3 (in the first column the results are averaged over 2000 instances, in second column over 1000, in the column third over 8000, and in the last column over 11000, and in all cases averaged overall values of d).

The results in Tables 2 and 3 show that the behavior of each heuristic algorithm w.r.t. the quality of the solution on the TM and the TS instances is essentially the same. We can note that when ANIS$\leq$ 2.5, the best solution output by algorithms M, MGH, $H_{W_1}^{MBS}$ is definitively better than ours, regardless of the number of vertices. On the other hand, the performance of the ICE algorithm improves by increasing n and ANIS. In particular, when ANIS$\geq$ 3 and $n \geq 500$ or when ANIS$\geq$ 3.5 and $n \geq 250$, our approach outperforms the best one among M, MGH, $H_{W_1}^{MBS}$.

6.2. Interval instances

In the present section we discuss the computational results obtained by running algorithms M, MGH, $H_{W_1}^{MBS}$, and ICE on instances with arbitrary interval conflict graphs. The test bed was generated as we now describe.

By $TI(n, B, \Delta)$ we denote a set of 100 randomly generated instances of $BPPC$ with $n \in \{120, 250, 500, 1000\}$ intervals, weights uniformly distributed in $[20, 100]$ (as in Falkenauer (1996)), bound $B \in \{120, 150, 180, 210, 240, 270, 300, 330, 360, 390\}$, and interval conflict graphs with expected edge density $\Delta \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. When $\Delta > 0$, we repeatedly run the random interval graph generator described in Section 5 and we selected 100 sets of

		ANIS	$\{2, 2.5\}$	3	$\{3.5, \ldots, 6.5\}$	$\{2, \ldots, 6.5\} \cup \{6.7\}$
		B	$\{120, 150\}$	180	$\{210, \ldots, 390\}$	$\{120, \ldots, 390\} \cup \{400\}$
n	120	MS=LB	10.00%	45.00%	87.88%	69.82%
		MS<ICE	79.50%	23.00%	3.12%	18.82%
		Gap_MS	4.26%	1.27%	0.33%	1.13%
		ICE=LB	3.50%	37.00%	86.12%	66.64%
		ICE<MS	0.50%	4.00%	0.88%	1.09%
		Gap_ICE	7.78%	1.70%	0.39%	1.85%
	250	MS=LB	7.50%	33.00%	82.12%	64.09%
		MS<ICE	91.50%	23.00%	1.50%	19.82%
		Gap_MS	3.52%	0.84%	0.27%	0.91%
		ICE=LB	3.00%	31.00%	85.75%	65.73%
		ICE<MS	0.00%	6.00%	4.62%	3.91%
		Gap_ICE	6.89%	1.04%	0.19%	1.49%
	500	MS=LB	2.00%	25.00%	73.12%	55.82%
		MS<ICE	92.00%	10.00%	0.75%	18.18%
		Gap_MS	2.99%	0.71%	0.26%	0.79%
		ICE=LB	0.00%	31.00%	81.38%	62.00%
		ICE<MS	2.00%	25.00%	12.00%	11.36%
		Gap_ICE	5.97%	0.61%	0.14%	1.24%
	1000	MS=LB	0.50%	20.00%	69.00%	52.09%
		MS<ICE	87.00%	6.00%	1.25%	17.27%
		Gap_MS	2.12%	0.60%	0.21%	0.60%
		ICE=LB	0.00%	29.00%	85.12%	64.55%
		ICE<MS	3.00%	52.00%	22.88%	21.91%
		Gap_ICE	4.96%	0.39%	0.05%	0.97%

Table 2: Aggregate computational results on $TM(n, B, f(d))$. In grey, the best data of a cell.

		ANIS	$\{2, 2.5\}$	3	$\{3.5, \ldots, 6.5\}$	$\{2, \ldots, 6.5\} \cup \{6.7\}$
		B	$\{3000, 3750\}$	4500	$\{5250, \ldots, 9750\}$	$\{3000, \ldots, 9750\} \cup \{10000\}$
n	120	MS=LB	8.00%	43.00%	84.75%	67.00%
		MS<ICE	78.50%	26.00%	2.62%	18.55%
		Gap_MS	4.82%	1.29%	0.48%	1.35%
		ICE=LB	3.00%	35.00%	84.38%	65.09%
		ICE<MS	0.50%	2.00%	1.62%	1.45%
		Gap_ICE	8.18%	1.83%	0.49%	2.01%
	250	MS=LB	5.00%	28.00%	81.75%	62.91%
		MS<ICE	89.00%	25.00%	1.88%	19.82%
		Gap_MS	3.62%	1.02%	0.29%	0.96%
		ICE=LB	1.50%	28.00%	84.12%	64.00%
		ICE<MS	0.00%	11.00%	4.00%	3.91%
		Gap_ICE	6.66%	1.26%	0.23%	1.49%
	500	MS=LB	3.00%	31.00%	73.38%	56.73%
		MS<ICE	94.00%	12.00%	1.12%	19.00%
		Gap_MS	2.80%	0.73%	0.27%	0.77%
		ICE=LB	0.50%	34.00%	83.88%	64.18%
		ICE<MS	1.00%	35.00%	14.12%	13.64%
		Gap_ICE	5.78%	0.61%	0.13%	1.20%
	1000	MS=LB	1.00%	26.00%	67.88%	51.91%
		MS<ICE	93.50%	8.00%	1.25%	18.64%
		Gap_MS	2.16%	0.68%	0.25%	0.64%
		ICE=LB	0.00%	28.00%	79.12%	60.09%
		ICE<MS	1.50%	53.00%	23.25%	22.00%
		Gap_ICE	5.13%	0.46%	0.08%	1.03%

Table 3: Aggregate computational results on $TS(n, B, f(d))$. In grey, the best data of a cell.

n intervals whose intersection graph had edge density $\delta \in [\Delta - 0.02; \Delta + 0.02]$. When $\Delta = 0$, we define the set $\mathcal{I} = \{I_h = (h, h+1), h = 0, \ldots, n-1\}$ of n mutually non-intersecting intervals (in this case $BPPC$ reduces to BP). Totally we build 40000 instances and test each algorithm on all of them.

The detailed results obtained for $n = 1000$ are presented in Table 4, where rows are indexed by Δ and columns by ANIS and B; the results for $n \in \{120, 250, 500, 1000\}$ are summarized in Table 5. We make use of the same notations described in Section 6.1.

Similarly to the instances with threshold conflict graphs and for the same reasons, the algorithms M, MGH, $H_{W_1}^{MBS}$ have essentially better results than ICE when ANIS ≤ 2.5. In this case, G_w is not an interval graph anymore.

On the other hand, differently from the results of the previous section, our algorithm outperforms the other ones on most of the instances with ANIS ≥ 3 and $\Delta \leq 0.7$. We believe that this is mainly due to the fact that, in PHASE I, our algorithm uses the informations of the interval model of G for finding a vertex coloring solution $\{V_1, V_2, \ldots, V_\lambda\}$ for the interval graph taking care that their weights $W(V_i)$, for $i = 1, 2, \ldots$, are as balanced as possible: for increasing ANIS, this turns out to be more and more an optimal solution for $BPPC$. On the contrary, the other algorithms do not take advantage of the order given by the endpoints of the intervals and base their choices on the vertex weight and on its degree in the graph, only. On threshold graphs too, these algorithms did not consider these informations, but the very particular structure of the threshold graphs and the presence of a unique maximal clique led them to better solutions.

The results show that our algorithm is strictly better than (equal to) the other algorithms on more than the 56% (25%) of total number of instances. The cyan cells in Table 4 show that algorithm ICE solves to optimality all the 100 instances of 37 out of 100 cells, while algorithms M, MGH, $H_{W_1}^{MBS}$ solve to optimality all the 100 instances of 14 out of 100 cells, precisely those with ANIS ≥ 3.5 and $\Delta \geq 0.8$.

As for the computing times, the comments given in Section 6.1 hold, except that the average times required by $H_{W_1}^{MBS}$ on one instance is 1.1 seconds, its maximum being 2.6 seconds on one instance when $B = 180$ and $\Delta = 0.3$.

In Table 5, the results for $n \in \{120, 250, 500, 1000\}$ are summarized (in the first column the results are averaged over 2000 instances, in second column over 1000, in the column third over 7000, and in the last column over 10000, and in all cases averaged overall values of d). According to the computational results, algorithm ICE is better than the other ones when ANIS$\geq$ 3.5 or when ANIS= 3 and $n \geq 500$. We also remark that for $B \in \{120, 150\}$ the values MS=LB and ICE=LB are very small: we suspect that this is due to the poor quality of LB_{BPPC} for ANIS ≤ 2.5 (Bacci (2018)).

7. Conclusions

In this paper we dealt with the Bin Packing Problem with Conflicts ($BPPC$) on instances with interval conflict graphs.

We proposed a two phases heuristic algorithm, which finds a weight balanced vertex coloring solution and possibly modifies it with a local search method.

We compared our heuristic algorithm with the best ones in the literature (Basnet and Wilson (2005); Fernandes Muritiba et al. (2010); Maiza and Radjef (2011)). Computational experiments are conducted by varying the number n of vertices, the edge density of the conflict graph, the value B, and the values of the weights, hence the average number of items per subset ANIS (we remark that the literature instances consider ANIS= 2.5, only). We considered both instances taken from the literature and randomly generated instances. The conflict graphs of the instances taken from the literature are very peculiar interval graphs, namely threshold graphs. On the other hand, to our knowledge, no random interval graph generator exists which outputs a graph with desired edge density. To this extent, we defined a new one with this properties which we

Δ		ANIS	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5
		B	120	150	180	210	240	270	300	330	360	390
	0.0	MS=LB	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
		MS<ICE	100%	45%	0%	0%	0%	0%	0%	0%	0%	0%
		Gap_MS	1.68%	1.19%	0.89%	1.48%	0.85%	0.94%	1.06%	0.68%	0.72%	0.77%
		ICE=LB	0%	0%	0%	19%	54%	78%	76%	81%	88%	87%
		ICE<MS	0%	24%	97%	100%	100%	100%	100%	98%	100%	99%
		Gap_ICE	7.76%	1.40%	0.42%	0.29%	0.18%	0.10%	0.12%	0.10%	0.07%	0.08%
	0.1	MS=LB	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
		MS<ICE	100%	54%	0%	0%	0%	0%	0%	0%	0%	0%
		Gap_MS	1.69%	1.27%	1.07%	1.68%	1.19%	1.41%	1.63%	1.65%	1.92%	2.20%
		ICE=LB	0%	0%	0%	6%	32%	59%	63%	71%	76%	82%
		ICE<MS	0%	21%	94%	100%	100%	100%	100%	100%	100%	100%
		Gap_ICE	7.82%	1.52%	0.58%	0.39%	0.28%	0.18%	0.18%	0.16%	0.14%	0.12%
	0.2	MS=LB	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
		MS<ICE	100%	44%	0%	0%	0%	0%	0%	0%	0%	0%
		Gap_MS	1.72%	1.59%	1.74%	2.39%	3.18%	3.95%	4.67%	5.88%	6.29%	5.91%
		ICE=LB	0%	0%	0%	0%	9%	18%	24%	19%	20%	8%
		ICE<MS	0%	32%	99%	100%	100%	100%	100%	100%	100%	100%
		Gap_ICE	7.93%	1.75%	0.72%	0.58%	0.44%	0.38%	0.40%	0.48%	0.52%	0.68%
	0.3	MS=LB	0%	0%	0%	0%	0%	0%	0%	1%	2%	2%
		MS<ICE	100%	34%	0%	0%	0%	0%	0%	0%	0%	0%
		Gap_MS	1.77%	2.60%	3.54%	4.63%	6.00%	5.78%	4.72%	4.72%	4.48%	4.41%
		ICE=LB	0%	0%	0%	0%	0%	0%	7%	59%	99%	100%
		ICE<MS	0%	57%	100%	100%	100%	100%	100%	98%	98%	98%
		Gap_ICE	7.92%	2.32%	1.21%	1.19%	1.26%	1.51%	1.57%	0.30%	0.01%	0.00%
	0.4	MS=LB	0%	0%	0%	0%	0%	1%	2%	2%	2%	2%
		MS<ICE	100%	26%	1%	3%	0%	1%	0%	0%	0%	0%
		Gap_MS	1.81%	4.20%	5.00%	4.50%	2.82%	2.28%	2.23%	2.22%	2.21%	2.20%
		ICE=LB	0%	0%	0%	0%	13%	85%	100%	100%	100%	100%
		ICE<MS	0%	63%	99%	89%	96%	99%	98%	98%	98%	98%
		Gap_ICE	8.40%	3.70%	2.79%	3.38%	1.32%	0.06%	0.00%	0.00%	0.00%	0.00%
	0.5	MS=LB	0%	0%	0%	1%	2%	2%	2%	2%	2%	2%
		MS<ICE	100%	89%	90%	2%	0%	0%	0%	0%	0%	0%
		Gap_MS	1.93%	4.84%	3.27%	1.20%	1.10%	1.09%	1.08%	1.08%	1.08%	1.08%
		ICE=LB	0%	0%	0%	38%	97%	100%	100%	100%	100%	100%
		ICE<MS	0%	8%	5%	93%	97%	98%	98%	98%	98%	98%
		Gap_ICE	10.17%	6.14%	4.41%	0.33%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%
	0.6	MS=LB	0%	0%	3%	17%	19%	20%	20%	20%	20%	20%
		MS<ICE	100%	97%	27%	0%	0%	0%	0%	0%	0%	0%
		Gap_MS	2.43%	4.26%	0.62%	0.35%	0.34%	0.34%	0.34%	0.34%	0.34%	0.34%
		ICE=LB	0%	0%	9%	98%	100%	100%	100%	100%	100%	100%
		ICE<MS	0%	2%	44%	83%	81%	80%	80%	80%	80%	80%
		Gap_ICE	12.41%	7.33%	0.54%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	0.7	MS=LB	0%	0%	46%	87%	87%	87%	87%	87%	87%	87%
		MS<ICE	100%	100%	17%	0%	0%	0%	0%	0%	0%	0%
		Gap_MS	6.02%	0.80%	0.14%	0.03%	0.03%	0.03%	0.03%	0.03%	0.03%	0.03%
		ICE=LB	0%	0%	65%	100%	100%	100%	100%	100%	100%	100%
		ICE<MS	0%	0%	37%	13%	13%	13%	13%	13%	13%	13%
		Gap_ICE	15.43%	2.90%	0.09%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	0.8	MS=LB	0%	3%	82%	100%	100%	100%	100%	100%	100%	100%
		MS<ICE	100%	96%	7%	0%	0%	0%	0%	0%	0%	0%
		Gap_MS	3.77%	0.42%	0.03%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		ICE=LB	0%	0%	89%	100%	100%	100%	100%	100%	100%	100%
		ICE<MS	0%	2%	15%	0%	0%	0%	0%	0%	0%	0%
		Gap_ICE	8.57%	1.27%	0.02%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	0.9	MS=LB	0%	12%	93%	100%	100%	100%	100%	100%	100%	100%
		MS<ICE	100%	87%	7%	0%	0%	0%	0%	0%	0%	0%
		Gap_MS	2.49%	0.25%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		ICE=LB	0%	2%	93%	100%	100%	100%	100%	100%	100%	100%
		ICE<MS	0%	10%	7%	0%	0%	0%	0%	0%	0%	0%
		Gap_ICE	4.78%	0.66%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		tmin_ICE	0.2902	0.1556	0.0875	0.0290	0.0292	0.0281	0.0273	0.0262	0.0262	0.0263
		tmax_ICE	4.9518	2.1335	1.1226	0.6880	0.4123	0.2697	0.2101	0.1714	0.1443	0.1265
		tavg_ICE	1.8321	0.7242	0.3558	0.2064	0.1294	0.0932	0.0744	0.0633	0.0554	0.0489

Table 4: Computational results on $TI(1000, B, \Delta)$. In grey, the best data of a cell; In cyan, the data of the algorithm with 100% of certified optimum.

			ANIS	{2, 2.5}	3	{3.5, . . . , 6.5}	{2, . . . , 6.5}
			B	{120, 150}	180	{210, . . . , 390}	{120, . . . , 390}
n	120	MS=LB		11.90%	44.30%	70.50%	56.16%
		MS<ICE		83.70%	18.10%	1.73%	19.76%
		Gap_MS		4.91%	2.24%	1.30%	2.12%
		ICE=LB		4.40%	42.30%	83.04%	63.24%
		ICE<MS		1.20%	14.20%	17.69%	14.04%
		Gap_ICE		9.26%	2.35%	0.68%	2.56%
	250	MS=LB		6.15%	34.20%	52.54%	41.43%
		MS<ICE		89.35%	17.60%	0.64%	20.08%
		Gap_MS		4.06%	2.06%	1.53%	2.09%
		ICE=LB		1.30%	32.30%	79.14%	58.89%
		ICE<MS		2.80%	25.40%	40.67%	31.57%
		Gap_ICE		8.18%	1.90%	0.48%	2.16%
	500	MS=LB		3.75%	27.50%	39.47%	31.13%
		MS<ICE		85.00%	15.90%	0.33%	18.82%
		Gap_MS		3.01%	1.89%	1.70%	1.98%
		ICE=LB		0.75%	27.80%	75.87%	56.04%
		ICE<MS		6.85%	46.10%	58.03%	46.60%
		Gap_ICE		6.85%	1.45%	0.35%	1.76%
	1000	MS=LB		0.75%	22.40%	31.03%	24.11%
		MS<ICE		83.60%	14.90%	0.09%	18.27%
		Gap_MS		2.34%	1.63%	1.63%	1.77%
		ICE=LB		0.10%	25.60%	73.80%	54.24%
		ICE<MS		10.95%	59.70%	68.60%	56.18%
		Gap_ICE		6.01%	1.08%	0.24%	1.48%

Table 5: Aggregate computational results on $TI(n, B, \Delta)$. In grey, the best data of a cell.

used to create our test bed.

The computational results show that our heuristic algorithm definitely outperforms the ones we considered when the average number of items per subset is greater than or equal to 3.5, both on instances with interval and threshold conflict graphs.

The main heuristic contribution from the literature are an adaptation to $BPPC$ of classical algorithms for the Bin Packing or output a feasible solution by repeatedly finding an independent subset of vertices. We think that the good performances of our algorithm depend on the fact that it starts with a Vertex Coloring solution, i.e. a family of independent subsets, and then locally rearranges them with the aim of obtaining subsets with feasible weight. Generally speaking, we believe that this approach may be successful also for solving $BPPC$ instances with arbitrary conflict graphs, especially when a good Vertex Coloring solution is easy to find.

References

T. Bacci. *On the Block Relocation Problem and the Bin Packing Problem with conflicts on interval graphs*. PhD thesis, University of Rome Tor Vergata, 2018.

T. Bacci and S. Nicoloso. On the benchmark instances for the Bin Packing with Conflicts, 2017. arXiv:1706.03526 [math.CO].

B. S. Baker and E. G. Coffman. Mutual Exclusion Scheduling. *Theoretical Computer Science*, 162:225–243, 1996.

C. Basnet and J. Wilson. Heuristics for determining the number of warehouses for storing non-compatible products. *International Transactions in Operations Research*, 12:527–538, 2005.

A. Bettinelli, V. Cacchiani, and E. Malaguti. Bounds and algorithms for the Knapsack problem with Conflict graph. Technical Report OR-14-16, DEIS, University of Bologna, July 2014.

H. L. Bodlaender and K. Jansen. Restrictions of graphs partition problem. Part I. *Theoretical Computer Science*, 148:93–109, 1995.

F. Brandão and J. P. Pedroso. Bin Packing and related problems: General arc-flow formulation with graph compression. *Computers & Operations Research*, 69:56–67, 2016.

R. Capua, Y. Frota, T. Vidal, and L. S. Ochi. Um algoritmo heurìstico para o problema de Bin Packing com Conflitos. SBPO'15, Porto de Galinhas, Brazil, August 25-28, 2015

N. Christofides. *Graph Theory: An Algorithmic Approach.* Academic Press, 1975.

N. Christofides, A. Mingozzi, and P. Toth. Loading problems. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 339–369. Wiley, Chichester, UK, 1979.

V. Chvátal and P. L. Hammer. Set-packing and threshold graphs. Technical Report Rept. CORR 73-21, University of Waterloo, 1973.

F. Clautiaux, A. Khanafer, S. Hanafi, and E. Talbi. Le problème bi-objectif de Bin-Packing avec Conflits. 12ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la décision ROADEF 2010, Saint-Etienne, France, Mar 2011.

D. Cornaz, F. Furini, and E. Malaguti. Solving Vertex Coloring problems as maximum weight Stable Set problems. *Discrete Applied Mathematics*, 217:151 – 162, 2017.

M. Delorme, M. Iori, and S. Martello. Bin Packing and Cutting Stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255:1–20, 2016.

S. Elhedhli, L. Li, M. Gzara, and J. Naoum-Sawaya. A Branch-and-Price algorithm for the Bin Packing problem with Conflicts. *INFORMS Journal on Computing*, 23:404–415, 2011.

L. Epstein and A. Levin. On Bin Packing with Conflicts. *SIAM Journal on Optimization*, 19: 1270–1298, 2008.

E. Falkenauer. A hybrid grouping genetic algorithm for Bin Packing. *Journal of Heuristics*, 2: 5–30, 1996.

F. Gardi. Mutual Exclusion Scheduling with interval graphs or related classes, part i. *Discrete Applied Mathematics*, 157:19–35, 2009.

M. R. Garey and D. S. Johnson. Strong NP-completeness results: motivation, examples, and emplications. *Journal ACM*, 25:499–508, 1978.

M. Gendreau, G. Laporte, and F. Semet. Heuristics and lower bounds for the Bin Packing problem with Conflicts. *Computers & Operations Research*, 31:347–358, 2004.

I. P. Gent. Heuristic solution of open Bin Packing problems. *Journal of Heuristics*, 3:299–304, 1998.

M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, 1980.

T. Gschwind and S. Irnich. Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, 28(1):175–194, 2016.

J. N. D. Gupta and J. C. Ho. A new heuristic algorithm for the one-dimensional Bin-Packing problem. *Production Planning & Control.*, 10(6):598–603, 1999.

R. Gupta, S. K. Bose, S. Sundarrajan, M. Chebiyam, and A. Chakrabarti. A two stage heuristic algorithm for solving the server consolidation problem with item-item and bin-item incompatibility constraints. In *Proceedings of the 2008 IEEE International Conference on Services Computing - Volume 2*, pages 39–46. IEEE Computer Society, 2008.

K. Jansen and S. Öhring. Approximation algorithms for time constrained scheduling. *Information and Computation*, 132(2):85 – 108, 1997.

D. S. Johnson. Fast algorithms for Bin Packing. *Journal of Computer and System Sciences*, 8: 272–314, 1974.

D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

C. Joncour. *Problèmes de placement 2D et application à l'ordonnancement: modélisation par la théorie des graphes et approches de programmation mathématique.* PhD thesis, Université de Bordeaux I, July 2010. Number 4173.

C. Joncour, S. Michel, R. Sadykov, D. Sverdlov, and F. Vanderbeck. Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics*, 36:695–702, 2010.

S. B. Jouida, A. Ouni, and S. Krichen. A multi-start tabu search based algorithm for solving the warehousing problem with conflict. In H. A. Le Thi, T. Pham Dinh, and N. T. Nguyen, editors, *Modelling, Computation and Optimization in Information Systems and Management Sciences: Proceedings of the 3rd International Conference on Modelling, Computation and Optimization in Information Systems and Management Sciences - MCO 2015 - Part II*, pages 117–128. Springer International Publishing, 2015.

J. Justicz, E. R. Scheinerman, and P. M. Winkler. Random intervals. *American Mathematical Monthly*, 97:881–889, 1990.

R. Kalfakakou, S. Katsavounis, and K. Tsouros. Minimum number of warehouses for storing simultaneously compatible products. *International Journal of Production Economics*, 81-82: 559–564, 2003.

A. Khanafer. *Algorithmes pour des problèmes de Bin Packing mono- et multi-objectif.* PhD thesis, Université Lille1, 2010. Number 40363.

A. Khanafer, F. Clautiaux, and E. Talbi. New lower bounds for Bin Packing problems with Conflicts. *European Journal of Operational Research*, 206(2):281–288, 2010.

A. Khanafer, F. Clautiaux, S. Hanafi, and E. Talbi. The min-conflict packing problem. *Computers & Operations Research*, 39:2122–2132, 2012a.

A. Khanafer, F. Clautiaux, and E. Talbi. Tree-decomposition based heuristics for the two-dimensional Bin Packing problem with conflicts. *Computers & Operations Research*, 39:54–63, 2012b.

D. Kowalczyk and R. Leus. An exact algorithm for parallel machine scheduling with conflicts. *Journal of Scheduling*, 2016. DOI: 10.1007/s10951-016-0482-0.

M. Maiza and C. Guéret. A new lower bound for Bin Packing problem with general conflicts graph. In *1st Doctoriales STIC'09*, Université de M'sila, Algérie, 2009.

M. Maiza and M. S. Radjef. Heuristics for solving the Bin-Packing problem with Conflicts. *Applied Mathematical Sciences*, 5:1739 – 1752, 2011.

S. Martello and P. Toth. Lower bounds and reduction procedures for the Bin Packing problem. *Discrete Applied Mathematics*, 28:59–70, 1990.

A. E. Fernandes Muritiba. *Algorithms and Models For Combinatorial Optimization Problems.* PhD thesis, Alma Mater Studiorum Università di Bologna, 2010.

A. E. Fernandes Muritiba, M. Iori, E. Malaguti, and P. Toth. Algorithms for the Bin Packing problem with Conflicts. In *XIII Workshop on Combinatorial Optimization*, 2009.

A. E. Fernandes Muritiba, M. Iori, E. Malaguti, and P. Toth. Algorithms for the Bin Packing problem with Conflicts. *INFORMS Journal on Computing*, 22(3):401–415, 2010.

Y.S. Myung. On the clique partitioning problem in weighted interval graphs. *Theoretical Computer Science*, 396:290 – 293, 2008.

R. Sadykov and F. Vanderbeck. Bin Packing with Conflicts: a generic Branch-and-Price algorithm. *INFORMS Journal on Computing*, 25(2):244–255, 2013.

I. Vasileios. Random interval graphs. Master's thesis, University of Essex, 2005.

Y. Yuan, Y. Li, and Y. Wang. An improved ACO algorithm for the Bin Packing problem with Conflicts based on graph coloring model. *21th International Conference on Management Science & Engineering, Helsinki, Finland*, 2014.