Assorted Load Balancing Schemes for Constrained Bin Packing Problems

Philippe Olivier[1,2], Andrea Lodi[1,2], and Gilles Pesant[1]

École Polytechnique de Montréal, Montreal, Canada[1]
CERC[2]
{philippe.olivier, andrea.lodi, gilles.pesant}@polymtl.ca

Abstract. Despite the existence of efficient solution methods for bin packing problems, in practice these seldom occur in such a pure form but feature instead various considerations such as pairwise conflicts or profits between items, or aiming for balanced loads amongst the bins. The Wedding Seating Problem is a combinatorial optimization problem incorporating elements of bin packing with conflicts, bin packing with profits, and load balancing. We use this representative problem to present and compare constraint programming and integer programming approaches.

1 Introduction

In the optimization version of the classical bin packing problem, a set of items of various weights must be packed into as few bins of limited capacities as possible. Despite the existence of efficient solution methods for bin packing problems, in practice these seldom occur in such a pure form. They instead feature various considerations such as pairwise conflicts or profits between items, or aiming for balanced loads amongst the bins. The objective then becomes to minimize some scoring function by selecting an optimal distribution of items in the available bins.

In our representative problem, the Wedding Seating Problem (WSP) [1], groups of guests of different sizes must be seated at tables of limited capacities. Some of these groups may or may not like each other, thus some relation is defined over each pair of them. Pairs of groups whose relation is *definitely apart* can never be seated at the same table. While not strictly necessary, pairs of groups whose relation is either *rather together* or *rather apart* should, if possible, be seated together or apart, respectively. Pairs which have no specific relation are *indifferent*. Note that an implicit relation, *definitively together*, is baked into the problem as *groups of guests*, the smallest indivisible entity that can be assigned to a table.

This paper is an extended and overhauled version of [**?**], which was presented at CPAIOR 2018. Section 2 gives a formal definition of our problem, and Sect. 3 reviews current methods of solving the WSP and similar problems. Sections 4 to 6 introduce, respectively, our constraint programming (CP) model and our two integer programming (IP) models. Section 8 presents the results of our experiments.

2 Description of the Problem

Let

- $\mathcal{I} = \{1, \ldots, n\}$ be the index set of items,
- $\mathcal{B} = \{1, \ldots, m\}$ be the index set of bins,
- ℓ and u be, respectively, the lower and upper bounds on the load of a bin,
- w_i denote the weight of item i with $w = \sum_{i \in \mathcal{I}} w_i$ representing the combined weight of all the items,
- c_{ij} the cost incurred if items i and j are packed into the same bin.

Entries in the cost matrix C can take any integer value. Namely,

$$c_{ij} \begin{cases} = \infty, & \text{if } i \text{ and } j \text{ are in conflict and must be packed into separate bins,} \\ = 0, & \text{if } i \text{ and } j \text{ have no cost,} \\ < 0, & \text{if } i \text{ and } j \text{ should rather be packed into the same bin,} \\ > 0, & \text{if } i \text{ and } j \text{ should rather be packed into separate bins.} \end{cases}$$

Since a conflict is expressed as being a prohibitive cost, the initial cost matrix can be enhanced by adding this prohibitive cost for each pair of items whose combined weights is greater than u, since they can never be packed together.

The problem consists of packing all items into the available bins such that conflicting items are packed into separate bins, under arbitrary load balancing constraints. The objective is to minimize the combined cost of all available bins. We will be considering the load balancing constraints under four distinct norms which yield incomparable results. The L_0-norm minimizes the number of values different from the mean bin load. To tackle fractional means with this norm, the two nearest integers will be considered to be within a valid mean range. The L_1- and L_2-norms minimize, respectively, the sum of absolute deviations and the sum of squared deviations from the mean. The L_∞-norm minimizes the maximum deviation from the mean.

In order to compare the impact of the load balancing parameters, we have opted to construct a Pareto set using the ϵ-constraint method [2]: We bounded the global deviation at successively higher intervals, each time solving the problem by minimizing the objective. We consider disjoint intervals of deviation to ensure nonoverlapping solution spaces in each step of the construction of the Pareto set, thus preventing identical solutions from being found in different steps. Parameters $d_{\min}$ and $d_{\max}$ denote the lower and upper bounds on the global deviation of a solution. Presenting the parameters of the load balancing constraints with a Pareto set has the advantage of offering decision-makers multiple optimal solutions to choose from depending on their perception of the trade-offs amongst them.

The problem of packing items into bins is the same as that of seating groups of guests at tables, as described by Lewis and Carroll. It has been shown to be $\mathcal{NP}$-hard as it generalizes two other $\mathcal{NP}$-hard problems: the k-partition problem and the graph k-coloring problem [3].

3 Related Work

The problem of constructing seating plans was originally introduced by Bellows and Peterson [4]. The authors used an IP model to solve various instances of this problem for their own wedding. They were able to solve small instances of 17 guests in a few seconds. For a larger instance of 107 guests, however, no optimal solution could be found in reasonable time.

This seating assignment problem was later formally described as the *Wedding Seating Problem* by Lewis [1]. The author solved the problem with a metaheuristic model based on a two-stage tabu search (of which our variant is discussed in Sect. 7). In a further paper [3], Lewis and Carroll devised their own quadratic IP model (of which our variant is discussed in Sect. 5) to be compared with their metaheuristic model. The latter approach outperformed the former both in solution quality and in running time in most cases. The authors also reimplemented the IP model of Bellows and Peterson, which they found performed poorly.

A fairly recent survey of CP work on bin packing and load balancing can be found in [5]. Most research on bin packing with conflicts, such as [6], focuses on minimizing the number of bins used as in the classical problem (albeit with the added conflict dimension). In contrast, the WSP uses a fixed number of bins of dynamic capacities, the objective being to optimize a scoring function subject to additional balancing constraints. The notion of pairwise costs between items used in the WSP is somewhat unconventional, but a similar idea can be found in some bin packing *games* where selfish agents (items) strive to maximize their payoffs by packing themselves into the most profitable bin, which is determined by its item composition [7].

4 CP Model

For each item i we define a decision variable b_i whose value is the bin in which the item will be packed. For each bin k we define an auxiliary variable o_k whose value is the load of the bin. To pack the items into the bins, the model uses a `binpacking` constraint [8]. The balancing objective represented by variable σ is taken care of by a `balance` constraint [9], which can handle L_1- and L_2-norms.

$$\texttt{binpacking}\,(\langle b_i\rangle, \langle w_i\rangle, \langle o_k\rangle) \tag{1}$$

$$\texttt{balance}\,(\{o_k\}, w/m, \sigma) \tag{2}$$

$$\ell \leq o_k \leq u \tag{3}$$

$$b_i \in \mathcal{B}, \quad i \in \mathcal{I} \tag{4}$$

$$\ell, u, o_k \in \mathbb{N}, \quad k \in \mathcal{B} \tag{5}$$

The model uses a conflict graph to infer `alldifferent` constraints, similar to what has been used by Gualandi and Lombardi in their decomposition of the `multibin packing` constraint [10]. In a conflict graph, each item is represented by a vertex, and an edge joins two vertices if they are conflicting. By extracting all

the maximal cliques of this graph, it is possible to add `alldifferent` constraints to the model for each one of those cliques. Furthermore, the maximum clique of an instance (the largest of all the maximal cliques) determines a lower bound on the number of bins necessary to find a feasible solution to that instance. The Bron-Kerbosch algorithm is an exact method which can be used to find all the maximal cliques of the conflict graph [11]. Let $\mathcal{M}$ be the set of all maximal cliques (maximal clique x being, for example, $\mathcal{M}_x = \{b_1^x, \ldots, b_k^x\}$):

$$\texttt{alldifferent}\left(\{\mathcal{M}_x\}\right), \quad \forall x \in \{1, \ldots, |\mathcal{M}|\} \tag{6}$$

While we have not explored all specific edge cases in this paper, if we were to solve highly constrained instances of the problem (i.e., with a dense conflict graph) these could be intractable for the Bron-Kerbosch algorithm. A heuristic for finding cliques could instead be applied, or simple binary disequality constraints could be used in lieu of the conflict graph.

Some symmetry is broken by fixing items of weights strictly greater than $u/2$ to separate bins. In theory, better symmetry breaking could be achieved first by fixing each item in the maximum clique of the conflict graph to a separate bin, and then by forcing an order on the loads of the remaining bins. In practice, however, symmetry breaking for the CP model is tricky as it interferes with the branching heuristic, whose strength lies in finding very good solutions very quickly. While the overall time needed to solve an instance to optimality decreases with the use of symmetry breaking, the downside is that early solutions will be worse with it than without. Without symmetry breaking, the branching heuristic basically packs the heaviest items at the bottom of the bins (i.e., they are assigned to a bin near the top of the tree). The top items of the bins are thus of lighter weights, and it is naturally less constraining to swap them around and pack them more profitably. Symmetry breaking forces some packings of lighter items at the bottom of the bins, constraining the swapping of items at the top of the bins.

Finally the objective is to minimize f subject to a constraint bounding the value of the global deviation of the bins:

$$d_{\min} < \sigma \leq d_{\max} \tag{7}$$

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} (b_i = b_j)\, c_{ij} \tag{8}$$

The model branches on the decision variables according to a heuristic which was inspired by the standard best fit decreasing strategy used to solve the bin packing problem. It first chooses the item of the greatest weight yet unassigned and will pack it into an empty bin, if one is available. Otherwise, the heuristic will pack the item into the bin which would most increase the f value. This heuristic tends to find a good first solution very early in the search.

We have also further tested the CP model by including a large neighborhood search (LNS) strategy [12]. The model finds a good initial solution, after which the LNS strategy takes over. The LNS will iteratively freeze different parts of the solution and search afresh from there for a set amount of time. About a third of the bins are frozen as such, with the most profitable bins having the most chance of being frozen.

5 IP Model A

The first IP model ($\mathrm{IP_A}$) is a generalization of the natural quadratic IP model proposed by Lewis and Carroll [3]. A barebones model defined by (9)–(19) is common to all load balancing norms. A $n \times m$ matrix of decision variables x represents packing assignments of items into bins (9)

$$x_{ik} = \begin{cases} 1, & \text{if item } i \text{ is packed into bin } k, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

$$\min \quad \sum_{k\in\mathcal{B}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ik}x_{jk}c_{ij} + \epsilon_1 \sum_{k\in\mathcal{B}} o_k + \epsilon_2\phi \tag{10}$$

s.t.

$$\sum_{k\in\mathcal{B}} x_{ik} = 1 \qquad \forall i \in \mathcal{I} \tag{11}$$

$$x_{ik} + x_{jk} \leq 1 \qquad \forall i,j \in \mathcal{I} : c_{ij} = \infty, \quad \forall k \in \mathcal{B} \tag{12}$$

$$\sum_{i\in\mathcal{I}} x_{ik}w_i \geq \ell \qquad \forall k \in \mathcal{B} \tag{13}$$

$$\sum_{i\in\mathcal{I}} x_{ik}w_i \leq u \qquad \forall k \in \mathcal{B} \tag{14}$$

$$x_{ik} = 0 \qquad \forall i \in \mathcal{I}, \quad \forall k \in \{i+1,\ldots,m\} \tag{15}$$

$$\sum_{i\in\mathcal{I}} x_{ik}w_{ik} - w/m \leq o_k \qquad \forall k \in \mathcal{B} \tag{16}$$

$$w/m - \sum_{i\in\mathcal{I}} x_{ik}w_{ik} \leq o_k \qquad \forall k \in \mathcal{B} \tag{17}$$

$$o_k \in \{\ell, ..., u\} \qquad \forall k \in \mathcal{B} \tag{18}$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in \mathcal{I}, \quad \forall k \in \mathcal{B} \tag{19}$$

The items are required to be packed into one and only one bin (11), and conflicting items may not be packed into the same bin (12). Constraints (13) and (14) require that the load of every bin be within bounds ℓ and u. Some symmetry breaking is achieved with (15). Constraints (16)–(18) emulate an absolute value

function to provide the absolute deviation of each bin. Finally, constraints (19) ensure the integrality of the solution. The barebones model minimizes the sum of all pairwise costs between items in each bin (10). The values of ϵ_1 and ϵ_2 are small enough that they do not impact the value of the objective by 1 or more, and are such that $\epsilon_1 \gg \epsilon_2$. They ensure that auxiliary variables o and ϕ correctly represent, respectively, the absolute deviation of the bins and the global deviation of the solution.

Integration of load balancing according to the L_0-norm is achieved with $\phi = (TODO)$, and by adding constraints (?)–(?) to the barebones model:

$$TODO \tag{20}$$

TODO: Description of the constraints.

Integration of load balancing according to the L_1-norm is achieved with $\phi = \sum_{k \in \mathcal{B}} o_k$, and by adding constraints (21) and (22) to the barebones model:

$$\sum_{k \in \mathcal{B}} o_k \geq d_{\min} \tag{21}$$

$$\sum_{k \in \mathcal{B}} o_k \leq d_{\max} \tag{22}$$

These constraints stipulate that the global deviation of the solution must be bounded by $d_{\min}$ and $d_{\max}$.

Integration of load balancing according to the L_2-norm is achieved with $\phi = \sum_{h \in \mathcal{B}} \sum_{k \in \mathcal{B}} y_{hk}$, and by adding constraints (23)–(29) to the barebones model (let $o_{\min} = 0$ and $o_{max} = \sqrt{d_{\max} \times (m-1)/m}$):

$$y_{hk} \geq o_{\min} o_k + o_h o_{\min} - o_{\min} o_{\min} \qquad \forall h, k \in \mathcal{B} \tag{23}$$

$$y_{hk} \geq o_{\max} o_k + o_h o_{\max} - o_{\max} o_{\max} \qquad \forall h, k \in \mathcal{B} \tag{24}$$

$$y_{hk} \leq o_{\max} o_k + o_h o_{\min} - o_{\max} o_{\min} \qquad \forall h, k \in \mathcal{B} \tag{25}$$

$$y_{hk} \leq o_h o_{\max} + o_{\min} o_k - o_{\min} o_{\max} \qquad \forall h, k \in \mathcal{B} \tag{26}$$

$$\sum_{h \in \mathcal{B}} \sum_{k \in \mathcal{B}} y_{hk} \geq d_{\min} \tag{27}$$

$$\sum_{h \in \mathcal{B}} \sum_{k \in \mathcal{B}} y_{hk} \leq d_{\max} \tag{28}$$

$$y_{hk} \geq 0 \qquad \forall h, k \in \mathcal{B} \tag{29}$$

Constraints (23)–(26) and (29) introduce a convex relaxation with McCormick envelopes [13]. Constraints (27) and (28) stipulate that the global deviation of the solution must be bounded by $d_{\min}$ and $d_{\max}$.

Integration of load balancing according to the L_∞-norm is achieved with $\phi = y$, and by adding constraints (30)–(32) to the barebones model:

$$o_k \leq y \qquad \forall k \in \mathcal{B} \tag{30}$$
$$y \geq d_{\min} \tag{31}$$
$$y \geq d_{\min} \tag{32}$$

Constraints (30) ensure that the maximal value of variables o is represented by y, and (31) and (32) stipulate that the global deviation of the solution must be bounded by $d_{\min}$ and $d_{\max}$.

6 IP Model B

Our second IP model (IP_B) is based on the definition of an exponential-size collection of possible bin compositions as for the classical bin packing problem. Indeed, as for bin packing, the resulting formulation can be solved by column generation with either a set covering (SC) or set partitioning (SP) model. Let $\mathbb{S}$ be the collection of all subsets of items that can be packed into the same bin

$$\mathbb{S} \coloneqq \left\{ S \subseteq \{1, \ldots, n\} : \ell \leq \sum_{i \in S} w_i \leq u, \quad \forall i, j \in S : c_{ij} \neq \infty \right\}.$$

We can observe that it may not be possible to *only* construct maximal sets with regards to the bin capacity due to conflicts between items and the constraints enforcing them. There is a binary variable for each subset $S \in \mathbb{S}$ representing a combination of items, or *pattern*, to be packed into the same bin

$$x_S \coloneqq \begin{cases} 1, & \text{if pattern } S \text{ is selected,} \\ 0, & \text{otherwise.} \end{cases} \tag{33}$$

The sum of all pairwise costs for the items of a pattern and the deviation of the weight of that pattern from the mean bin load are represented by α and β, respectively. In regards to the balancing objective, using a fixed number of bins has two major advantages over using a variable number of bins. First, the values of α and β need to be computed only once per pattern (when it is generated) and remain constant throughout the process. Second and more importantly, since β is computed outside of the program, the norm according to which it is computed does not complexify the problem (i.e., the program remains linear even when balance is computed according to an L_2-norm).

While the solution of a SP model is always directly feasible, that of a SC model must be transformed in order to be feasible for our problem (i.e., we must remove all duplicate items from the bins). This has the unfortunate effect of potentially worsening the objective funtion. Example 1 illustrates the underlying issue of using a SC model to solve our problem.

Example 1. Assume an instance of the problem with 2 bins and 4 items A, B, C, and D. The pairwise costs of AB, BC, CD, and DA are 0, while the pairwise

cost of AC is 1 and that of BD is -2. We also have $\ell = 2$ and $u = 3$. The two most profitable maximal subsets are ABD and BCD which both have a value of -2 and cover all the items. The initial solution of the SC model would be $(-2) + (-2) = -4$, which must be rendered feasible for our problem by removing B from one bin and D from the other. This modified solution would have a value of $(0) + (0) = 0$, while the solution of a SP model would be to pack AC and BD in separate bins, for a value of $(1) + (-2) = -1$.

The master problem (MP) is thus based on a SP model.

$$\min \quad \sum_{S \in \mathbb{S}} \alpha_S x_S \tag{34}$$

s.t.

$$\sum_{S \in \mathbb{S}: i \in S} x_S = 1 \qquad \forall i \in \mathcal{I} \tag{35}$$

$$\sum_{S \in \mathbb{S}} x_S = m \tag{36}$$

$$\sum_{S \in \mathbb{S}} \beta_S x_S \geq d_{\min} \tag{37}$$

$$\sum_{S \in \mathbb{S}} \beta_S x_S \leq d_{\max} \tag{38}$$

$$x_S \in \{0, 1\} \qquad \forall S \in \mathbb{S} \tag{39}$$

The MP minimizes the combined costs of all bins (23), under the conditions that each item be packed into one and only one bin (24), that a total of m bins be used (25), and that the global deviation of a solution be within bounds $d_{\min}$ and $d_{\max}$ (26)-(27). In order to begin solving the problem, the column generation algorithm needs m columns making up an initial feasible solution of the problem and of the continuous relaxation obtained by replacing constraints (28) with $x_S \geq 0, \forall S \in \mathbb{S}$. These columns are generated by a compact CP model defined by (1)-(5), (7), (todo: and some symmetry breaking and simple search strategy), and

$$b_i \neq b_j, \quad b_i, b_j \in \mathcal{B}, \quad \forall i, j \in \mathcal{I} : c_{ij} = \infty \tag{40}$$

which searches for the first feasible solution safisfying these constraints. The m columns found by this CP model make up the initial restricted master problem (RMP). The dual of the MP is

$$\max \quad \sum_{i=1}^{n} y_i + m\zeta + d_{\min}\gamma + d_{\max}\delta \tag{41}$$

s.t.

$$\sum_{i \in S} y_i + \zeta + \beta_S(\gamma + \delta) \leq \alpha_S \qquad \forall S \in \mathbb{S} \tag{42}$$

$$y_i \text{ free} \qquad \forall i \in S \tag{43}$$

$$\zeta \text{ free} \tag{44}$$

$$\gamma \geq 0 \tag{45}$$

$$\delta \leq 0 \tag{46}$$

which is all that is needed for the column generation algorithm to take over:

1. Solve the continuous relaxation of the RMP to get the dual values.
2. Solve the subproblem, or pricing problem (PP), to generate $S^* \subseteq \{1, \ldots, n\}$ (the most promising new column). Let

$$z_i := \begin{cases} 1, & \text{if item } i \text{ is packed into the new bin/pattern,} \\ 0, & \text{otherwise.} \end{cases} \tag{47}$$

The pricing problem is now solved by a CP model and is about 10 times faster. Average results when solving an instance of 50 items: L_1-norm: time=5.7s, LB=-132, absolute gap=3; L_2-norm: time=2.6s, LB=-128, absolute gap=1.75. I still need to do the L_0- and L_∞-norms.

$$\max \quad \sum_{i=1}^{n} y_i^* z_i + \zeta^* + \beta(\gamma^* + \delta^*) - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} z_i z_j \quad \text{if } \gamma^* + \delta^* < 0 \tag{48}$$

$$\max \quad \sum_{i=1}^{n} y_i^* z_i + \zeta^* - \beta(\gamma^* + \delta^*) - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} z_i z_j \quad \text{if } \gamma^* + \delta^* > 0 \tag{49}$$

s.t.

$$z_i + z_j \leq 1 \qquad \forall i, j \in \mathcal{I} : c_{ij} = \infty \tag{50}$$

$$\sum_{i=1}^{n} w_i z_i \geq \ell \tag{51}$$

$$\sum_{i=1}^{n} w_i z_i \leq u \tag{52}$$

$$\sum_{i=1}^{n} w_i z_i - w/m \leq \beta \tag{53}$$

$$\sum_{i=1}^{n} w_i z_i - w/m \geq -\beta \tag{54}$$

$$z_i \in \{0, 1\} \qquad \forall i \in \mathcal{I} \tag{55}$$

The PP minimizes the value of a bin (48)/(49) under the conditions that no conflicting items be packed into it (50), and that its load be within bounds

ℓ and u (51)-(52). Constraints (53)-(54) work in the same manner as constraints (15)-(16) and ensure that deviation β is always positive.

3. Determine if S^* should be added to the RMP. If the inequality $\sum_{k \in S^*} y_k^* + \zeta^* + \beta(\gamma^* + \delta^*) > \sum_{i \in S^*} \sum_{j \in S^*} c_{ij}$ (or $\sum_{k \in S^*} y_k^* + \zeta^* - \beta(\gamma^* + \delta^*) > \sum_{i \in S^*} \sum_{j \in S^*} c_{ij}$, alternatively) is true, compute α_{S^*} and β_{S^*}, and add column S^* to the RMP before going back to step 1. Otherwise, the current solution of the continuous relaxation of the RMP is the lower bound of the initial problem.

The optimal solution of the continuous relaxation of the RMP provides a lower bound to the problem. We subsequently enforce constraints (28) and solve the RMP with the columns which were previously generated by the algorithm in order to find an integral solution. We know such an integral solution exists since we started off with one with our initial columns. While this final integral solution offers no proof of optimality, it is most often relatively close to the lower bound.

Depending on the norm used to balance the bins and on the value of bound $d_{\max}$, we can make arithmetic deductions to determine the optimal ℓ and u bounds which should help prematurely prune nodes leading to infeasible solutions, without eliminating any feasible solution:

This will be updated to include the other norms.

$$\ell \geq \max\{0, \lceil w/m - d_{\max}/2 \rceil\} \tag{56}$$

$$u \leq \lfloor w/m + d_{\max}/2 \rfloor \tag{57}$$

$$\ell \geq \max\{0, \left\lceil w/m - \sqrt{d_{\max} \times (m-1)/m} \right\rceil\} \tag{58}$$

$$u \leq \left\lfloor w/m + \sqrt{d_{\max} \times (m-1)/m} \right\rfloor \tag{59}$$

For the L_1-norm, in the worst case a single bin can account for at most half of the deviation, since the cumulative weight in excess of the mean bin load will always be equal to the cumulative weight short of the mean bin load (56)-(57). For the L_2-norm we must also take the number of bins into account in order to tightly bound the most deviative bin (58)-(59). This offline optimization of bin load bounds makes a noticeable difference for both IP models, cutting the execution time by half on average. This optimization is of no use for the CP model, as the balancing constraint already ensures that bin loads be consistent with $d_{\max}$.

7 MH Model

I propose to construct the metaheuristic model proposed by Lewis (tabu search) from scratch. It will be designed in line with our other models (i.e., the objective function only takes costs into account, and we build solutions with $d_{\min}$ and $d_{\max}$), take into account all the norms, construct a Pareto front, etc.

8 Benchmark Results

I propose to remove the plain CP results and just keep the CP+LNS results, and mention textually what kind of improvement LNS offers.

Our testbed includes instances of 25 and 50 items, of weights chosen uniformly at random from 1 to 8 (in a similar fashion as Lewis [1]). Costs and conflicts are introduced according to probability p of a pairwise negative cost, probability q of a pairwise positive cost, and probability r of a conflict (i.e., $p+q+r \leq 1$, with the remainder being the probability that the pair incurs no cost). Costs range from -5 to 5, and the number of bins has been chosen so that the mean load is closest to 10.

The experiments were performed on dual core AMD 2.2 GHz processors with 8 GB of RAM running CentOS 7, and all models were constructed with IBM ILOG CPLEX Optimization Studio v12.5. Pareto sets are constructed for the smallest integral ranges of deviations (i.e., min/max deviations of 0/1, 1/2, ..., 19/20). Time limits cover only one range of deviations, meaning that the results for one method, for one figure, involve solving 20 independent problems. This also means that construction of the Pareto sets is easily parallelizable and can be scaled to limited resources by modifying its resolution (e.g., if 4 instances can be solved in parallel, a lower-resolution Pareto set can be constructed with min/max deviations of 0/5, 6/10, 11/15, 16/20). Each data point shows the average results of 5 different cost matrices applied to the same set of items. All figures show the results for instances of 25 or 50 items with the deviations computed according to an L_1-norm. For Figs. 1-3 the cost probabilities are $p = 0.25$, $q = 0.25$, and $r = 0.25$.

Fig. 1. Instances of 25 items with a time limit of 600s.

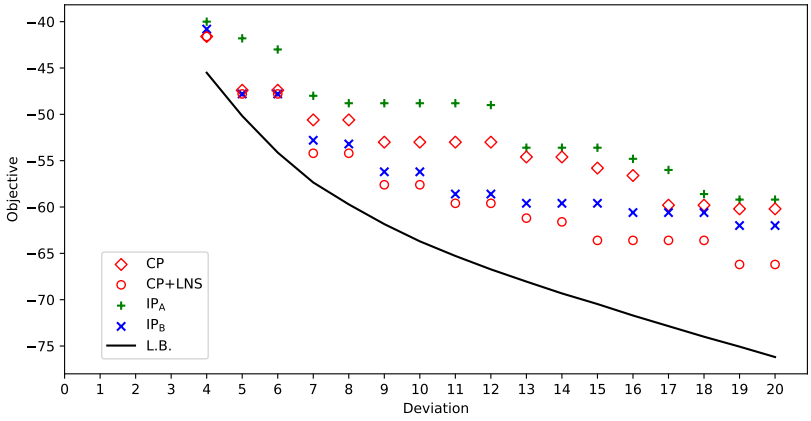


Fig. 2. Instances of 50 items with a time limit of 6s.

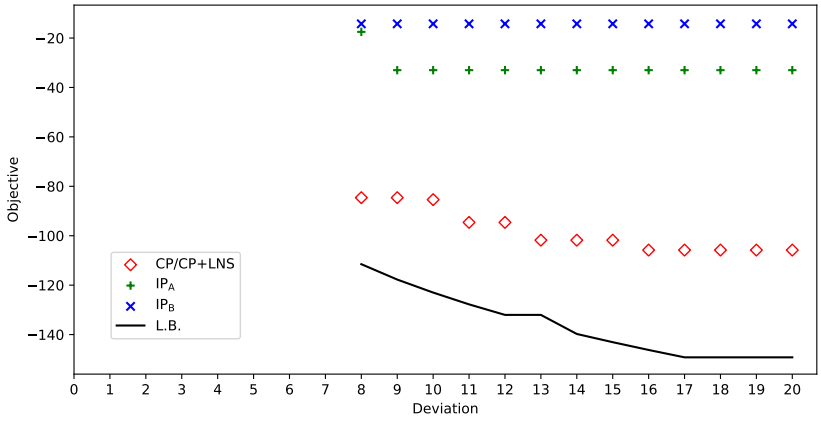
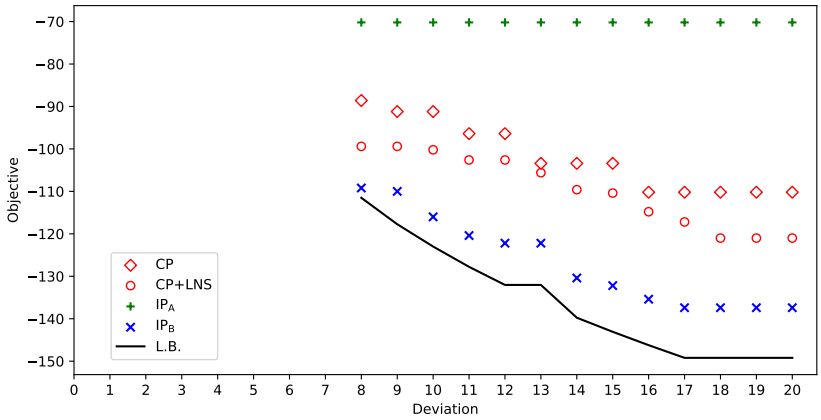


Fig. 3. Instances of 50 items with a time limit of 600s.



We can observe in Fig. 2 that the CP model finds the best early solutions of all methods. However it quickly reaches a plateau from which it is hard to further improve (notice the similarity between the CP solutions of Fig. 2 with a time limit of 6s. and those of Fig. 3 with a time limit of 600s.). The introduction of LNS for the CP model always improves the solution quality. For the small instances of Fig. 1, the CP model with LNS does better than IP_B since the latter only solves the relaxation of the RMP to optimality and tries to get the best integral solution from these limited columns, with no proof of optimality.

IP_A does particularly poorly compared with the other models. The results of IP_B are usually off to a slow start, but given enough time this model does better than both previous ones. Similar to the CP model, this model reaches a plateau of its own before the 600s mark. Further improvements could be achieved via branch and price. Average computation times are shown in Table 1 (owing to details in the implementation of the models, the execution time may be slightly higher than the time limit in some cases).

Table 1. Average computation times for Figs. 1-3

	25 items	50 items	
	600s.	6s.	600s.
CP	510.00	3.90	390.00
CP+LNS	508.49	3.90	379.02
IP_A	589.06	6.96	602.04
IP_B	8.35	8.32	178.46

For the CP and IP_A models, infeasibility or optimality can sometimes be proven when the maximum global deviation is low enough. IP_B does well especially when the time limit is high, although its solutions cannot be proven optimal without a branch-and-price scheme (but the CP model generating its initial columns can determine if an instance is infeasible). We have further tried solving instances with conflicts only ($p = 0$, $q = 0$, $r = 0.25$) and every method could find a solution within the time limit. In Fig. 4 we show the results of instances without conflicts and only with costs ($p = 0.25$, $q = 0.25$, $r = 0$).

Fig. 4. Instances of 50 items with a time limit of 600s. (costs only)

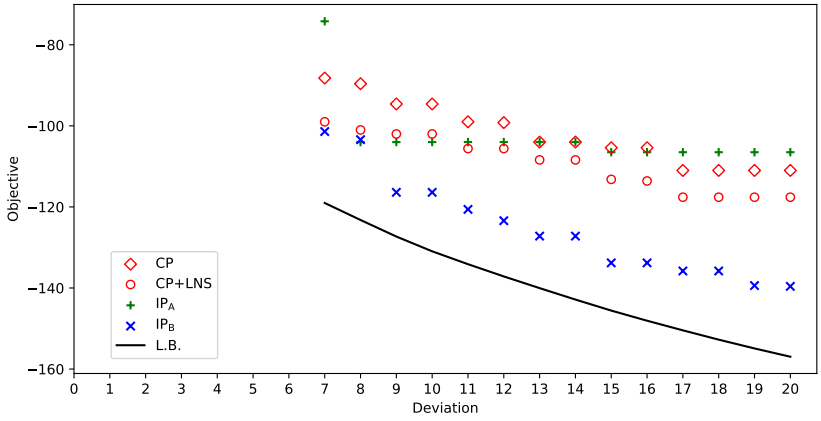


Table 2. Average computation times

	50 items	
	Conflicts only	Costs only
CP	0.03	420.00
CP+LNS	0.03	390.78
IP_A	601.11	600.62
IP_B	4.64	28.65

It is interesting to notice that with only conflicts, the CP model very easily proves optimality for all instances in a fraction of a second, whereas both IP models are orders of magnitude behind it. The weakness of the CP model lies in the optimization of objective f, even with LNS. IP_A appears to generally do better without conflicts, while the performance and results of IP_B are largely unaffected by the parameters of the instances.

A CP/IP_B hybrid could be constructed: The CP part would generate the initial columns, those columns being the current CP solution once the model reaches its plateau; From there, the IP part would take over and improve this solution by bringing it to near-optimality. Experimentation would be necessary to see if such a model would be an improvement over current methods. We have not integrated the CP and IP_B models into a hybrid in this paper, as our objective was to compare individual methods to each other. Furthermore, our simple approach to the generation of the Pareto sets for both IP models could be improved [14].

9 ~~Practical Applications~~

I am thinking of removing this section and instead mentioning this application in another section. The objective of the metaheuristic model (as it is found on the website) is not similar to that of our models (cost+deviation in its objective, instead of only cost for our objectives, with the deviation as a constraint). Furthermore it is poorly coded (it does not take into account the negative costs, which it should) and it is not flexible (costs are either -1, 0, or 1, while we can test our own models with whatever range of costs we like). Finally, it only takes into account the L_1-norm. So I suggest removing this section, re-coding the metaheuristic model to our standards, and including it as another model alongside our CP and 2 IP models. This would thereby add a new contribution to this paper, by introducing new balancing schemes for a metaheuristic model.

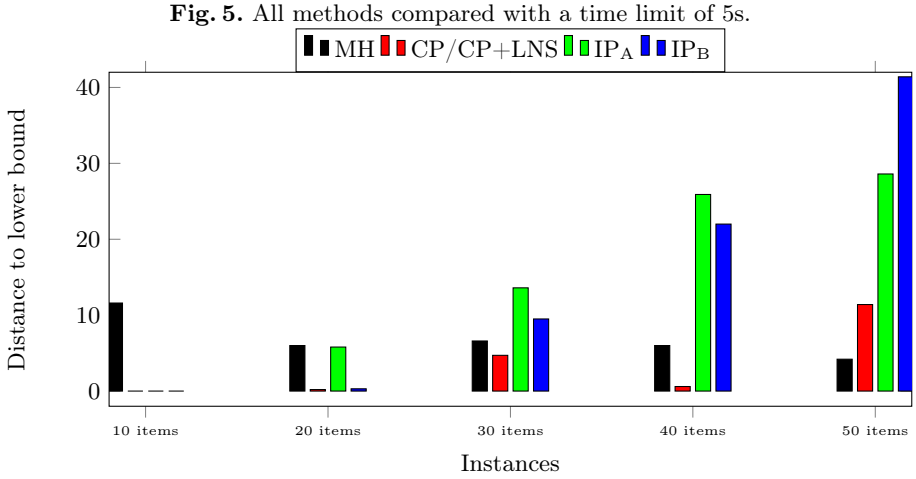
~~The metaheuristic approach developed by Lewis~~ [1] ~~is used on the commercial website `www.weddingseatplanner.com` as a tool to generate seating plans. The problem is similar to ours which is described in Sect. 2 of this paper, with a few exceptions: Bin loads are unbounded, negative costs are always equal to -1 and positive costs are always equal to 1, and the deviation is computed according to~~

an L_1-norm and directly added to the objective (i.e., the objective is to minimize f plus the deviation). The objective functions of our models have been adapted for these tests.

One of the design goals of the website was to solve the problem *quickly* since their clients could easily grow impatient after waiting just a few seconds in front of a seemingly unresponsive browser window. As such, their tool usually solves the problem in less than 5 seconds, a time limit which cannot be specified by the user. Because of this short time limit, we have bounded the maximum deviation of our models at 20 in order to find better results. For the CP model, since the balancing constraint and the branching heuristic are very effective, we have further decided to solve the instances two more times by bounding the maximum deviation at 10 and 5, respectively (since we are solving thrice as many problems, we have also divided the time limit by three). We will be considering only the best of those three solutions.

Due to an error in the website's implementation of the algorithm, all negative costs are considered to be conflicts. In order to provide a fair comparison, the instances we have generated for these tests do not include negative costs. The cost probabilities are thus $p = 0$, $q = 0.\bar{3}$, and $r = 0.\bar{3}$ (which implies that the probability of a cost of 0 is also $0.\bar{3}$). Since the website always solves the instances in under 5 seconds, this is what we have chosen as our time limit. Tests have been run with 10, 20, 30, 40, and 50 items, and are again averaged for five cost matrices. The histograms shown in Fig. 5 represent the distance in score of a solution from the lower bound.

Fig. 5. All methods compared with a time limit of 5s.



When solving small instances, exact methods have a distinct advantage over metaheuristics, often proving optimality. Both IP models scale poorly with an

~~increasing number of items as they usually require some time to find decent solutions. While it can find the best solutions given enough time, IP_B does particularly badly with a short time limit as the quality of its solutions improves relatively slowly. The metaheuristic model scales very well, with its solution quality being constant with a varying number of items. The CP model does well all-around, proving optimality for small instances as well as having good solutions for all instances.~~

10 Conclusion

In this paper we have compared how various methods can be used to solve multi-objective constrained bin packing problems with an aspect of load balancing. A metaheuristic model can find good solutions in a short time and scales well to an increasing number of items but will most likely not find optimal solutions. A CP model can also find good solutions quickly, but for large instances it will not reach the best solutions in reasonable time even with the help of LNS. A natural IP model is probably not the best choice, as it scales poorly while its strenghts can also be found in other models. An IP model using column generation does very well given enough time but is not a good contender to solving instances quickly. It would be interesting to see if a CP/IP hybrid using column generation and branch and price could prove optimality in reasonable time for larger instances of the problem.

Acknowledgements

Financial support for this research was provided by NSERC Discovery Grant 218028/2017 and CERC, École Polytechnique de Montréal.

References

1. Lewis, R. In: Constructing Wedding Seating Plans: A Tabu Subject. CSREA Press (2013) 24–32
2. Miettinen, K.: Nonlinear Multiobjective Optimization. Springer US (1998)
3. Lewis, R., Carroll, F.: Creating seating plans: a practical application. Journal of the Operational Research Society **67**(11) (Nov 2016) 1353–1362
4. Bellows, M.L., Peterson, J.D.L.: Finding an optimal seating chart (02 2012)
5. Schaus, P.: Solving Balancing and Bin-Packing problems with Constraint Programming. PhD thesis, Université catholique de Louvain (2009)
6. Sadykov, R., Vanderbeck, F.: Bin Packing with conflicts: a generic branch-and-price algorithm. INFORMS Journal on Computing **25**(2) (2013) 244–255
7. Wang, Z., Han, X., Dósa, G., Tuza, Z. In: Bin Packing Game with an Interest Matrix. Springer International Publishing, Cham (2015) 57–69
8. Shaw, P.: A constraint for bin packing. In Wallace, M., ed.: Principles and Practice of Constraint Programming – CP 2004, Berlin, Heidelberg, Springer Berlin Heidelberg (2004) 648–662

9. Pesant, G.: Achieving domain consistency and counting solutions for dispersion constraints. INFORMS Journal on Computing **27**(4) (2015) 690–703
10. Gualandi, S., Lombardi, M. In: A Simple and Effective Decomposition for the Multidimensional Binpacking Constraint. Springer Berlin Heidelberg, Berlin, Heidelberg (2013) 356–364
11. Bron, C., Kerbosch, J.: Algorithm 457: Finding all cliques of an undirected graph. Commun. ACM **16**(9) (September 1973) 575–577
12. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In Maher, M., Puget, J.F., eds.: Principles and Practice of Constraint Programming — CP98, Berlin, Heidelberg, Springer Berlin Heidelberg (1998) 417–431
13. Mitsos, A., Chachuat, B., Barton, P.I.: Mccormick-based relaxations of algorithms. SIAM Journal on Optimization **20**(2) (2009) 573–601
14. Boland, N., Charkhgard, H., Savelsbergh, M.: A criterion space search algorithm for biobjective integer programming: The balanced box method. INFORMS Journal on Computing **27**(4) (2015) 735–754