Process Plant Layout Optimization: Equipment Allocation

Gleb Belov[1], Tobias Czauderna[1], Maria Garcia de la Banda[1], Matthias Klapperstueck[1], Ilankaikone Senthooran[1], Mitch Smith[2], Michael Wybrow[1], and Mark Wallace[1]

[1] Monash University, Faculty of Information Technology, Australia
{name.surname}@monash.edu
[2] Woodside Energy Ltd., Australia

Abstract. Designing the layout of a chemical plant is a complex and important task. Its main objective is to find a most economical spatial arrangement of the equipment and associated pipes that satisfies construction, operation, maintenance and safety constraints. The problem is so complex it is still solved manually, taking multiple engineers many months (or even years) to complete. This paper provides (a) the most comprehensive model ever reported in the literature for spatially arranging the equipment, and (b) a Large Neighbourhood Search framework that enables complete solvers explore much larger neighbourhoods than previous approaches to this problem. The two contributions are part of a system being developed in collaboration with Woodside Energy Ltd. for arranging their Liquefied Natural Gas plants. The results are indeed so promising that Woodside are actively exploring its commercialisation.

1 Introduction

A chemical process plant produces chemicals by transforming or separating materials as they pass through different equipment via connecting pipes [10]. These plants are common in many industries, such as oil and gas, and are very costly to design, build and maintain, requiring multibillion-dollar budgets. When designing the layout of a new plant, the objective is to find a most economical spatial arrangement of the equipment and associated pipes that satisfies construction, operation, maintenance, and safety constraints.

High-quality layout can have a very significant impact on the cost of these plants. It can considerably reduce the cost of the pipes and associated support structures, which are known to take the largest share: up to 80% of the purchased equipment cost or 20% of the fixed-capital investment [12]. It also greatly reduces the total amount of space/volume needed, which is crucial for offshore plants. However, finding high-quality plant layouts is remarkably difficult due to the size of these plants and the complexity of the associated constraints. As a result, layouts are still designed manually, taking multiple engineers many months (or even years) to complete. This process is inefficient, costly and the results may vary in quality, since they largely depend on the experience of the piping and layout engineers. For this reason, Woodside Energy Ltd. funded our project to explore the use of optimisation and visualisation technology in improving the current layout design process for their Liquefied Natural Gas plants.

Due to the complexity of this problem, most methods published in this area divide it into independently solved two phases: the first phase spatially places the equipment, while the second routes the connecting pipes. However, these methods (e.g., [6, 14, 17, 18]) are too simplistic to meet industry requirements and/or do not scale. Our work aims to produce a comprehensive solution that satisfies real-world needs. We reported a model for the pipe routing phase of the problem in [3]. This paper presents two further contributions. The first one is the most realistic model ever reported for solving the equipment allocation phase. Prior to this, the most complete approach [6] only considered non-overlapping constraints and the minimisation of approximate pipe lengths, elevation and footprint. In addition to these, our model handles constraints on equipment alignment, simplified maintenance access, and support structures. While vital to model the real problem, all this (particularly the maintenance constraints) considerably increases the complexity of the model and, thus, decreases the scalability of the approach for complete search methods. The second contribution helps in this regard: a Large Neighbourhood Search [16] (LNS) framework that uses a modified neighbourhood definition and warm-start for several complete solvers (via the MiniZinc [11] modelling language). The resulting framework is highly efficient and can explore larger neighbourhoods than the only previous LNS approach [18] for this problem.

The model and LNS framework are parts of a much larger system being developed with Woodside Energy Ltd. This system aims at transforming the way in which Woodside engineers approach plant-layout design, by allowing them to get a global view of the plant layout quickly, and easily compare different design decisions. As our experimental evaluation shows, our approach provides us with the solution quality and scalability required for this application. The results are so promising Woodside is actively exploring commercialisation of the system.

2 Literature Review

To solve the plant layout problem we are required to find 3D location coordinates for the equipment and connecting pipes within a plant's volume (referred to as the *container space*), in such a way as to minimise the total cost of the plant while, at the same time, ensuring its safety and correct functionality. For small problem instances, one can apply integrated approaches (e.g., [14]) that simultaneously place the equipment and route the pipes. For larger, more realistic instances, current integrated approaches do not scale. For example, [14] fails to find any solution for plants with just 10 pieces of equipment and 8 pipes.

As a result, methods to solve the plant layout problem often divide it into two phases [6]. The first one finds the position and orientation of each piece of equipment, minimising an approximate total cost for the plant. The second phase looks for an optimal pipe routing to connect the already positioned equipment. There has been a significant amount of research devoted to the first phase (e.g., [6, 14, 17, 18]), which is the focus of our paper. However, most methods are too simplistic, concentrating mainly on (often 2D) non-overlapping constraints. No method we know of considers maintenance access constraints, alignment constraints, or the cost of (not) using provided support structures for elevated equipment.

In addition, scalability is a difficult issue due to the high combinatorial nature of the problem, given by the packing requirements. Approaches that use complete methods (e.g., [6, 14]) quickly become prohibitive as the number of objects grow. Others achieve scalability by giving up on completeness and, for example, iteratively constructing layouts that extend an initial, incomplete plant with more and more equipment [17], or using a quadratic force minimisation model of the problem that tries to compress an initial (complete) layout by minimizing "forces" between connected pieces of equipment [9]. As near-optimality is important for Woodside, we looked at LNS solutions to the problem and only found [18], which uses LNS to solve a simple 2D version of the problem.

The closest commercial product for plant layout is ASD Global's OptiPlant [1], which performs automated pipe routing and can generate an initial 3D plant layout from an equipment list. However, this product only deals with phase two (pipe routing), requiring users to specify equipment positions.

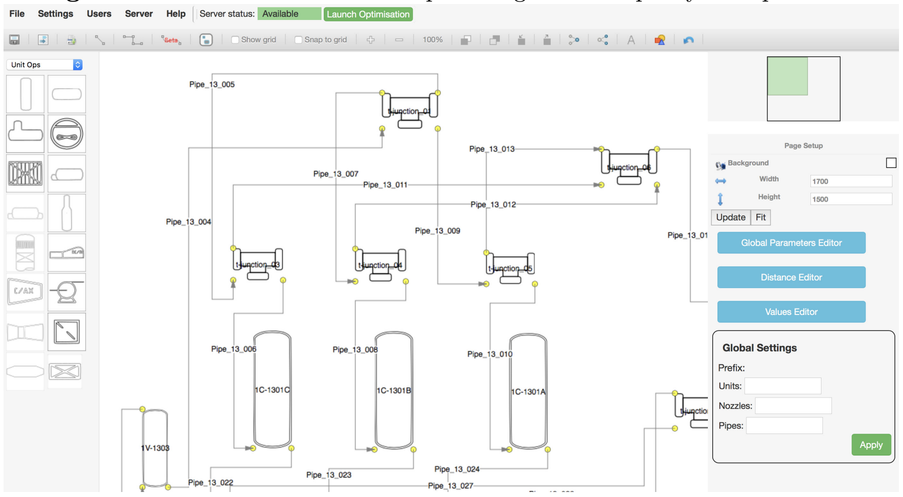
3 Full System and Key Role of Constraint Programming

As mentioned before, this paper describes components of a much larger system being developed in collaboration with Woodside Energy Ltd. The system has a 2D visual interface (see Figure 1) for users to specify the input data needed for the optimisation process. This information is stored in a JSON file and, when users press the "Launch Optimisation" button, is sent to a C++ program that generates the MiniZinc model for the first phase (described in this paper). MiniZinc compiles this model to the target solver and executes it to obtain the final positions and rotations of the equipment, writing these as additional entries in JSON format. Then, the C++ program uses the stored solution to generate a MiniZinc model for the second phase (described in [3]), compiles this to the target solver and executes it to obtain the routing of every pipe, again storing this in JSON format.

Once the equipment is placed and the pipes routed, a Python extension to FreeCAD generates a 3D model of the solution with structural information for navigating the equipment and pipes in the plant. An interactive 3D visualisation of the 3D model (see Figure 2) enables engineers to explore the produced layout collaboratively, evaluate and validate the proposed solution in a familiar way, and compare different solutions. Now that our LNS framework can produce high quality solutions in a relatively short amount of time (see Section 6), the aim is to support engineers in interactively re-optimising a given solution.

The use of a constraint programming modelling language (MiniZinc) has been critical to the success of our project for three main reasons. First, it has allowed us to quickly modify the models and try different modelling alternatives. This

Fig. 1. Process Editor interface for plant engineers to specify all input data.



Server status: Available

Width: 1700

Height: 1500

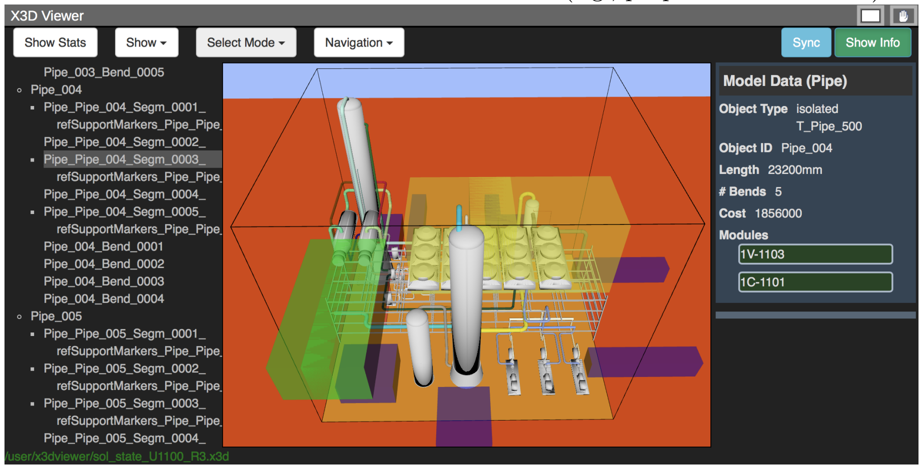
has been invaluable as our lack of knowledge regarding chemical plants often created issues that were only resolved by constantly exploring model changes with Woodside's engineers. Further model changes occurred as Woodside's engineers became more familiar with the possibilities offered by optimisation technology and we could increase the amount and quality of the constraints. Second, it has allowed us to easily compare the efficiency of different solvers. This has been particularly important since, as the models evolved, the efficiency of the solvers varied greatly. Being able to quickly determine which solver is faster for the current model allows us to give Woodside's engineers the best efficiency. Finally, it has allowed us to quickly build a system that far exceeds the capabilities of those currently available both in the literature and commercially. As a result, Woodside engineers can now obtain one or more near-optimal layouts for a plant significantly faster than before, and can already explore the consequences of applying simple modifications to the data (e.g., costs and safety distances).

4 An Optimization Model for Equipment Allocation

The model provided in this paper describes the first phase of the process plant layout problem. That is, determining the 3D position coordinates and the orientation of the given equipment within a given *container space*, that (a) satisfy safety, maintenance and alignment constraints and (b) minimise the costs of the land needed (the *footprint*), the supporting equipment, and the connecting pipes (see Figure 2 for a final solution to our small benchmark).

Our model considers the container space as a discretized cuboid with front-left-bottom (FLB) corner at coordinates $(0, 0, 0)$, and a (maximum) user-defined length, width and height corresponding to axes x, y and z, respectively, where $x \times$

Fig. 2. Visualisation of the final layout for Unit 1000 as a 3D model. Coloured boxes indicate different kinds of maintenance access zones (e.g., purple are truck access).



Model Data (Pipe)

Object Type isolated T_Pipe_500
Object ID Pipe_004
Length 23200mm
# Bends 5
Cost 1856000
Modules
1V-1103
1C-1101

y defines the footprint. The exact shape of each piece of *equipment* is abstracted by a bounding *box*, also represented by the position of its FLB corner and its user-defined (fixed) length, width and height. Boxes are positioned into one of four different *horizontal rotations* around the vertical axis: $0°, 90°, 180°$ and $270°$.

Each *nozzle* connecting a pipe with diameter d to equipment b at a position with centre $\overline{p} \in \mathbb{Z}^3$ relative to the equipment box's FLB corner, is represented as a point located $3d$ units perpendicularly away from the surface containing $\overline{p}$. This roughly corresponds to the usual bend position of a nozzle segment. Every pipe has a user-defined cost factor (per length of unit) that determines its cost. In this phase, pipe length is approximated by the Manhattan distance obtained from the position of its two connecting nozzles. The splitting of a pipe into two (or merging of two pipes into one) is modelled by a special rotatable *t-junction* box that inherits the pipe properties (e.g., size and safety distances).

User-defined *safety distances* between all equipment must be enforced. Default safety distances are specified between equipment *classes* (each equipment belongs to a class), and can be modified by providing a specific safety distance between any two. Note that safety distances are *directed*: that from A to B, where A is before B w.r.t. their projection on axis c, might be different than that from B to A. This might be useful for expressing vertical relative positions, for example, between fin-fans and other equipment, as well as horizontal positions from high-risk equipment in areas with well-defined wind directions.

Some equipment (such as the pipe *rack*) can support other equipment without further capital costs. Thus, equipment might be positioned on the ground (at no cost), on some other equipment (again at no cost), or "in the air", representing the fact that a supporting structure must be built. The cost of this structure is approximated using the cost (per height unit) associated to each piece of equipment (see *bph* below). Some equipment is allowed to protrude by a given amount (its *support margin*) over the sides of its support box. If the amount is negative, the box should be that far inside the support box's sides.

Some equipment should be located at a certain minimal level above the baseline of another. For example, a vessel might need to be a certain distance above a pump, if its flow into the pump needs to satisfy the *Net Positive Suction Head (NPSH)* regulation. We formulate the corresponding constraint in terms of minimal height differences. Note that they cannot be expressed by safety distances, as the size of the lower object might be larger than the elevation distance.

Some equipment has particular *maintenance access* requirements, such as the need to be accessible from above/below, or by a truck (requiring a big empty space to be attached to it and accessible from the road). The former is modelled by a constraint that ensures no other equipment is positioned above or below. The latter is modelled using additional (slave) boxes that satisfy the requirements in terms of size and location relative to the (master) equipment boxes. Currently, our model handles three types of relative locations:

- *Rigid (fixed) attachment*: slave is at a specified position relative to the master and the whole combination rotates together. Examples: access must be provided to a particular side of the equipment.
- *Rotatable attachment*: as before, but slave can rotate around the master. Example: access must be provided to any side of the equipment.
- *Multi-zone disjunctive attachment*: slave located in one of the given zones and orientated as the master. Example: aligning a pipe header to a pump group can be modelled by attaching one or several zone boxes to the pump group and requiring the header to be in one of these zones.

The rest of the section provides a summary of the parameters (input and derived data), variables, constraints and objective function used in our model.

4.1 Input and Derived Data

Input Index Sets.

- $\mathcal{B} = \{1, \ldots, N\mathcal{B}\}$: set of boxes that need to be allocated.
- $\mathcal{P} = \{1, \ldots, N\mathcal{P}\}$: set of pipes that connect the equipment.
- $\mathcal{MAZ} \subset \mathcal{B}$: subset of boxes that are maintenance access zones.
- $\mathcal{B}^{Supp} \subseteq \mathcal{B} \cup \{N\mathcal{B}+1\}$: subset of boxes that can support other boxes without further construction costs ($N\mathcal{B}+1$ represents the ground).
- $\mathcal{ATTZN} \subset \mathcal{B}$: subset of multi-zone attachment zones.

We currently assume there are two nozzles per pipe, and use input set $\mathcal{P}$ to construct the set $\mathcal{NZ} = \{(p,k) | p \in \mathcal{P}, k \in \{1,2\}\}$ of nozzles. We also use index set $\mathcal{OH} = \{1,2,3,4\}$ to represent rotations $\{0°, 90°, 180°, 270°\}$, respectively.

Input Data.

- $\overline{W}_i^0 \in \mathbb{Z}^{\times 3}$: x, y and z sizes of box $i \in \mathcal{B}$ while in horizontal rotation 0°.

- $\overline{XFLB}_i^{\mathrm{LB}}, \overline{XBRT}_i^{\mathrm{UB}} \in \mathbb{Z}^3$: lower bound of the FLB corner of box $i \in \mathcal{B}$, and upper bound of its back-right-top (BRT) corner, respectively. Often determined by the container space.
- $\mathcal{BOH}_i \subseteq \mathcal{OH}$: set of allowed horizontal orientations for box $i \in \mathcal{B}$.
- $bsd^H_{i,j}, bsd^V_{i,j} \in \mathbb{Z}$: horizontal and vertical directed safety distances, respectively, between boxes $i, j \in \mathcal{B}$.
- $\mathcal{NPSH}$: set of tuples (i, j, h), where $i, j \in \mathcal{B}$ and $h \in \mathbb{Z}$, indicating that the base of box j must be above the base of i by at least h units.
- $suppMrg_i \in \mathbb{Z}$: support margins for box $i \in \mathcal{B}$.
- $bph_i \in \mathbb{Z}$: height support penalty for box $i \in \mathcal{B}$, in \$ per height unit.
- $batt_i \in \{0, 1, 2, 3\}$: attachment type of box $i \in \mathcal{B}$, where 0 indicates none, 1 rigid, 2 rotatable, and 3 multi-zone.
- $batm_i \in \mathcal{B}$: attachment master box of slave box $i \in \mathcal{B}$ ($batm_i = i$ if $batt_i = 0$).
- $\overline{batp}_i \in \mathbb{Z}^3$: attachment point of box $i \in \mathcal{B}$ relative to its master's FLB corner in orientation $0°$, for rigid and rotatable attachment types.
- $\mathcal{B}atz_i \subset \mathcal{B}$: set of possible location zones (i.e., boxes, typically from $\mathcal{ATTZN}$) for the FLB corners of slave box $i \in \mathcal{B}$ with multi-zone attachment.
- $nzBox_i \in \mathcal{B}$: master box of nozzle $i \in \mathcal{NZ}$.
- $\overline{nzPoz}_i^0 \in \mathbb{Z}^3$: position of nozzle $i \in \mathcal{NZ}$ in its master box (orientation $0°$).
- $plc_i \in \mathbb{Z}$: cost factor for pipe $i \in \mathcal{P}$, in \$ per length unit.
- $fpc_c \in \mathbb{Z}$, $c \in \{x, y\}$: cost factor for perimeter length and width, respectively, in \$ per length unit.

4.2 Decision Variables

A solution to an instance of our model is expressed in terms of the values of the following decision variables (the first two groups functionally define all others):

- $r_i^F \in \mathcal{BOH}_\rangle$: final orientations of each box $i \in \mathcal{B}$.
- $\overline{XFLB}_i, \overline{XBRT}_i = \overline{XFLB}_i + \overline{W} \in \mathbb{Z}^3$ positions of the FLB and BRT corners of each box $i \in \mathcal{B}$, respectively.
- $\overline{W}_i^F \in \mathbb{Z}^3$: final sizes of each box $i \in \mathcal{B}$ according to its final rotation.
- $relPos_{ijc} \in \{0, 1\}$ relative position variable (directed separation flag) for pair of boxes $i, j \in \mathcal{B}$ along coordinate axis $c \in \{x, y, z\}$. It holds: $relPos_{ijc} = 1$ *iff* box j is after i in projection on axis c, obeying their safety distance.
- $suppIdx_i \in \mathcal{B}^{Supp}$: supporting box of each box $i \in \mathcal{B}$ s.t., $bph_i > 0$.
- $suppCost_i \in \mathbb{Z}$: computed support cost for each box $i \in \mathcal{B}$.
- $\overline{nzPos}_i \in \mathbb{Z}^3$: absolute position of each nozzle $i \in \mathcal{NZ}$.
- $\overline{pLen}_i \in \mathbb{Z}^3$: approximated length along each axis, for each pipe $i \in \mathcal{P}$.
- $fps_c \in \mathbb{Z}$, $c \in \{x, y\}$: footprint length and width, respectively.
- $obj \in \mathbb{Z}$: objective function value.

4.3 Functions

The constraints in our model use the following five functions. Function getBoxSafety : $\mathcal{B} \times \mathcal{B} \times \{x, y, z\} \to \mathbb{Z}$ returns the minimal positive separation from box $i \in \mathcal{B}$

to box $j \in \mathcal{B}$ along axis c:

$$\text{getBoxSafety}(i,j,c) = \begin{cases} bsd^H_{i,j}, & c \text{ is x or y} \\ bsd^V_{i,j}, & c \text{ is z} \end{cases} \tag{1}$$

Function hFindFLBCorner : $\mathbb{Z}^3 \times \mathcal{OH} \to \mathbb{Z}^3$ returns the new position of the FLB corner of a box with sizes $\overline{W} = (W_x, W_y, W_z)$, once it is rotated according to r. It uses the *element* constraint [2] to select a matrix column using r as index:

$$\text{hFindFLBCorner}(\overline{W}, r) = \begin{pmatrix} 0 & W_y & W_z & 0 \\ 0 & 0 & W_y & W_x \\ 0 & 0 & 0 & 0 \end{pmatrix}_{\cdot r} \tag{2}$$

Note that the above function and several of the constraints defined later (e.g., (9) and (13b)) are non-linear and, thus, not directly supported by MIP solvers. The MIP interface of MiniZinc [4] handles their MIP decomposition.

Function hRotateB : $\mathbb{Z}^3 \times \mathcal{OH} \to \mathbb{Z}^3$ returns the new sizes of a box with sizes $\overline{W} = (W_x, W_y, W_z)$, rotated according to r.

$$\text{hRotateB}(\overline{W}, r) = \begin{pmatrix} W_x & W_y & W_x & W_y \\ W_y & W_x & W_y & W_x \\ W_z & W_z & W_z & W_z \end{pmatrix}_{\cdot r} \tag{3}$$

Function hRotateBWB : $\mathbb{Z}^3 \times \mathbb{Z}^3 \times \mathbb{Z}^3 \times \mathcal{OH} \to \mathbb{Z}^3$ receives the sizes $\overline{W}_s$ and $\overline{W}_m$ of boxes s and m, respectively, the point $\overline{P}$ where the FLB corner of s is rigidly attached to m (relative to m's FLB corner, which is always (0,0,0)) in their default orientation, and rotation r. Returns the relative position of the FLB corner of s to that of m, once both are rotated by r around m's centre.

$$\text{hRotateBWB}(\overline{W}_s, \overline{W}_m, \overline{P}, r) = \begin{pmatrix} P_x & W_{my} - P_y - W_{sy} & W_{mx} - P_x - W_{sx} & P_y \\ P_y & P_x & W_{my} - P_y - W_{sy} & W_{mx} - P_x - W_{sx} \\ P_z & P_z & P_z & P_z \end{pmatrix}_{\cdot r} \tag{4}$$

Function hRotateBAB : $\mathbb{Z}^3 \times \mathbb{Z}^3 \times \mathbb{Z}^3 \times \mathcal{OH} \times \mathcal{OH} \to \mathbb{Z}^3$ receives the sizes $\overline{W}_s$ and $\overline{W}_m$ of boxes s and m, respectively, the point $\overline{P}$ where the FLB corner of s is attached to m (relative to m's FLB corner, which is always (0,0,0)) in their default orientation, and two rotations r_s and r_m. For efficiency, it returns an approximation of the relative position of the FLB corner of s relative to m, once s and m are rotated according to r_s and r_m, respectively, around the centre of m. The approximation is done by "shrinking" the master box to a square footprint, which is acceptable as long as we attach spacious MAZ.

$$\text{hRotateBAB}(\overline{W}_s, \overline{W}_m, \overline{P}, r_s, r_m) = \begin{pmatrix} \lfloor W^{r_m}_{mx}/2 \rfloor - \lceil w/2 \rceil \\ \lfloor W^{r_m}_{my}/2 \rfloor - \lceil w/2 \rceil \\ +0 \end{pmatrix} + \\ \text{hRotateBWB}\left(\overline{W}_s, \begin{pmatrix} w \\ w \\ W_{sz} \end{pmatrix}, \overline{P} + \begin{pmatrix} \lceil w/2 \rceil - \lfloor W_{mx}/2 \rfloor \\ \lceil w/2 \rceil - \lfloor W_{my}/2 \rfloor \\ 0 \end{pmatrix}, r_s \right) \tag{5}$$

where $w = \min\{W_{mx}, W_{my}\}$ is the minimum of the master's horizontal sizes and $\overline{W}^{r_m}_m = \text{hRotateB}(\overline{W}_m, r_m)$ are the sizes of the master rotated according to r_m.

Function hRotatePWB : $\mathbb{Z}^3 \times \mathbb{Z}^3 \times \mathcal{OH} \to \mathbb{Z}^3$ receives a rotation r and a point $\overline{P}$ rigidly attached to a box with sizes $\overline{W}$. Returns the relative position of $\overline{P}$ to the box's FLB corner, once both are rotated by r around the box's centre.

$$\text{hRotatePWB}(\overline{P}, \overline{W}, r) = \text{hRotateBWB}(\overline{0}, \overline{W}, \overline{P}, r) \tag{6}$$

4.4 Constraints and Objective Function

Box sizes: can be obtained from the original sizes and the final rotations:

$$\overline{W}_b^F = \text{hRotateB}(\overline{W}_b^0, r_b^F), \qquad b \in \mathcal{B} \tag{7}$$

Box position: should satisfy the given bounds.

$$\overline{XFLB}_b^{\text{LB}} \leq \overline{XFLB}_b, \quad \overline{XRBT}_b \leq \overline{XRBT}_b^{\text{UB}}, \qquad b \in \mathcal{B} \tag{8}$$

Box disjointness: only needs to be enforced between boxes that are not maintenance access zones (which are allowed to overlap), and are not attached to each other. If we had equal safety distances, we could have enforced disjointedness by using the `diffn_k` global constraint [2]. Since this is not the case, we enforce the disjointness of boxes $i, j \in \mathcal{B}$ similarly to [6], as follows. First, we reify the existence of the appropriate safety distance from i to j in each axis c as:

$$relPos_{ijc} = 1 \;\leftrightarrow$$
$$\begin{cases} True, & i = j \;\vee\; \{i,j\} \subseteq \mathcal{MAZ} \\ & \vee\; i \text{ and } j \text{ are attached} \\ & \vee\; i \in \mathcal{ATTZN} \vee j \in \mathcal{ATTZN} \\ \overline{XBRT}_{ic} + \text{getBoxSafety}(i,j,c) \leq \overline{XFLB}_{jc}, & \text{otherwise,} \end{cases}$$
$$i, j \in \mathcal{B},\; c \in \{x, y, z\} \tag{9}$$

Then, for each pair of boxes $i < j \in \mathcal{B}$, we demand the existence of such safety distance in at least one coordinate direction, positive or negative:

$$\bigvee_{c=1}^{3} \left(relPos_{ijc} \vee relPos_{jic}\right), \qquad i < j \in \mathcal{B} \tag{10}$$

This allows us to easily model the "none above/below" constraints by providing big enough vertical separations (ensuring they do not fit above each other).

Minimal height separation constraints: they demand the base of box j to be above the base of box i by at least h units:

$$\overline{XFLB}_{iz} + h \leq \overline{XFLB}_{jz}, \qquad (i,j,h) \in \mathcal{NPSH} \tag{11}$$

Support constraints: Box $i \in \mathcal{B}$ with positive height support cost ($bph_i > 0$) is considered as being supported by another box $j \in \mathcal{B}^{Supp} \setminus \{N^{\mathcal{B}} + 1\}$ (not the ground) if i's "core footprint" is contained in the footprint of j:

$$suppIdx_i = j \;\leftrightarrow$$
$$\left\{ \overline{XFLB}_{ic} + suppMrg_i \geq \overline{XFLB}_{jc} \wedge \overline{XBRT}_{ic} - suppMrg_i \leq \overline{XBRT}_{jc} \quad \right\},$$
$$c \in \{x,y\}, i \in \mathcal{B}, bph_i > 0,\; j \in \mathcal{B}^{Supp} \setminus \{N^{\mathcal{B}} + 1\} \tag{12}$$

The chosen form of the constraint implies that the support objects cannot be stacked if they already support something (as $suppIdx_i$ can only have one value). For efficiency, the domain of $suppIdx_i$ should be a priori reduced if possible. Finally, the support costs penalize being higher than the support object:

$$suppCost_i \geq 0, \tag{13a}$$

$$suppCost_i \geq bph_i \left(\overline{XFLB}_{iz} - (\overline{XFLB}_{\cdot z} +\!\!+ (0))_{suppIdx_i}\right), \quad i \in \mathcal{B} : bph_i > 0 \tag{13b}$$

which assumes the ground at level 0. This allows the modelling of objects that overlap with their supports, as it is the case with the equipment placed in racks.

Box attachment and MAZ: For a rigidly attached box (attachment type $batt_b = 1$), its FLB corner is computed from that of the master and the position of the attachment point, once rigidly rotated with the master to its final position.

$$\overline{XFLB}_b = \overline{XFLB}_{batm_b} + \text{hRotateBWB}(\overline{W}^0_b, \overline{W}^0_{batm_b}, \overline{batp}_b, r^F_b), \quad \forall b : batt_b = 1 \tag{14a}$$

Moreover its final orientation must be equal to that of its master $batm_b$:

$$r^F_b = r^F_{batm_b}, \qquad b \in \mathcal{B} : batt_b = 1 \tag{14b}$$

For rotatable, attached boxes ($batt_b = 2$), where the master has a square footprint, the modelling is the same as (14a). For rotation around a non-square-footprint master, we only provide an approximation (see hRotateBAB):

$$\overline{X}_b = \overline{X}_{batm_b} + \text{hRotateBAB}(\overline{W}^0_b, \overline{W}^0_{batm_b}, \overline{batp}_b, r^F_b, r^F_{batm_b}),$$
$$b : batt_b = 2,\ \overline{W}^0_{batm_b,1} \neq \overline{W}^0_{batm_b,2} \tag{15}$$

For multi-zone attachment ($batt_b = 3$), we require the slave's FLB corner to be in one of the specified zones (boxes in $\mathcal{B}atz_b$) and the rotation to be the same as that of the master ($batm_b$). This translates into the following system:

$$r^F_b = r^F_{batm_b}, \tag{16a}$$

$$\exists i \in \mathcal{B}atz_b : \overline{XFLB}_b \in [\overline{XFLB}'_i, \overline{XFLB}'_i + \overline{W}_i], \qquad b : batt_b = 3 \tag{16b}$$

where

$$\overline{XFLB}'_i = \overline{XFLB}_i + \text{hFindFLBCorner}(\overline{W}^0_b, r^F_b), \qquad i \in \mathcal{B}atz_b \tag{16c}$$

are the location zones' origins corrected for the slave's rotation.

Pipe symmetry. A set of pipes $\mathcal{S} \subset \mathcal{P}$ might need to be symmetric, e.g., due to restrictions on the associated equipment. Our phase one model approximates pipe symmetry by demanding the nozzle distances $\overline{pLen}_p$ of all pipes $p \in \mathcal{S}$ to be equal. Actual symmetry is enforced during pipe routing in the second phase.

Pipe cost approximation: uses the nozzle positions, which are computed using its master box's position and orientation:

$$\overline{nzPos}_{(p,i)} = \overline{XFLB}_{nzBox_{(p,i)}} +$$
$$\text{hRotatePWB}\left(\overline{nzPos}^0_{(p,i)}, \overline{W}^0_{nzBox_{(p,i)}}, r^F_{nzBox_{(p,i)}}\right),\ p \in \mathcal{P},\ i \in \{1, 2\} \tag{17}$$

This allows us to compute the pipe end differences and their absolute values:

$$\overline{pLen}_p = |\overline{nzPos}_{(p,2)} - \overline{nzPos}_{(p,1)}|, \qquad p \in \mathcal{P} \qquad (18)$$

Footprint cost approximation: similar to [6], we approximate footprint cost by penalizing perimeter length. We measure the footprint as including all physical boxes (i.e., no MAZ and zones) and, for each direction x or y, we only include boxes whose corresponding coordinate is not a priori fixed, as follows:

$$fps_c = \max\{\overline{XFLB}_{bc} | b \in \mathcal{BF}_c\} - \min\{\overline{XBRT}_{bc} | b \in \mathcal{BF}_c\},$$
$$\text{where} \quad \mathcal{BF}_c = \{b \in \mathcal{B} | \overline{XFLB}_{bc} \not\equiv \text{const}\} \setminus \mathcal{MAZ} \setminus \mathcal{ATTZN}, \ c \in \{x, y\}$$

Objective function: sum of the piping, footprint and support costs.

$$obj = \textstyle\sum_{p \in \mathcal{P}, c \in \{x,y,z\}} plc_p pLen_{pc} + \sum_{b \in \mathcal{B}} suppCost_b + \sum_{c \in \{x,y\}} fpc_c \cdot fps_c \quad (19)$$

Optimisations: We reduce the domain of each $suppIdx_i$ by removing boxes that are either too small to support box i, or (for a given neighbourhood) known to be located in a different area of the plant. We remove any non-overlapping constraints for pairs of boxes known not to overlap (for a given neighbourhood).

5 Overall Approach and Implementation

LNS is a meta-heuristic search method that, from an initial *seed* solution, iteratively relaxes part of the current solution and re-optimises the corresponding sub-problem obtaining a new solution. Our implementation uses a C++ program that, in each iteration, creates a new MiniZinc model for the neighbourhood, compiles it and executes it with a CP or MIP solver, using warm-starts when possible. The following describes these steps in more detail.

Constructing a seed solution: In order to have some control over the amount of time invested in finding a seed solution, we take two steps. First, we run the chosen solver until a feasible solution is found. Then we warm-start the same solver with that solution and run it with a given time limit.

Selecting the boxes to be relaxed: When relaxing a solution, and given input parameters $L \leq U$, our LNS first selects a subset of boxes $\mathcal{B}^{\mathcal{NBH}} \subset \mathcal{B}$ that will be relaxed (i.e., allowed to move freely) as follows: First, as long as we have not yet selected L boxes, a new box i is selected from set $\mathcal{B}$ and added to set $\mathcal{B}^{\mathcal{NBH}}$. For sequential LNS, i is a next biggest equipment box (starting from a new one for each $\mathcal{B}^{\mathcal{NBH}}$); for random LNS, i is selected randomly. Next, all slave boxes attached to i and all boxes connected to i via pipes are added to $\mathcal{B}^{\mathcal{NBH}}$, stopping if its cardinality reaches U. We repeat the process until the minimum number L of boxes is reached. This is different from [18] which constructs $\mathcal{B}^{\mathcal{NBH}}$ by choosing the boxes based on various probabilistic selection schemes, from random to those considering the number of box connections, the cost of the box, or all boxes connected to the selected links.

Defining the neighbourhood: Once $\mathcal{B}^{\mathcal{NBH}}$ is selected, we define the *neighbourhood of the current solution* (subset of solutions to be explored in this iteration) by fixing the orientations of all boxes not in $\mathcal{B}^{\mathcal{NBH}}$, and tightening their separation constraints (10). The tightening is done for all pairs of boxes $i, j \in \mathcal{B} \setminus \mathcal{B}^{\mathcal{NBH}}$ such that $i < j$, by enforcing their relative position along one of the six directions where they were most separated in the last solution. That is, by setting $relPos_{abc} = 1$ for one of the six tuples in $\{(a, b, c) | \{a, b\} = \{i, j\}, c \in \{x, y, z\}\}$, one with the maximum value for $\overline{XFLB}_{bc} - \overline{XBRT}_{ac} - \text{getBoxSafety}(a, b, c)$. The other five relative position variables (and, consequently, their reification constraints (9)) for i, j are omitted, significantly simplifying the model and enhancing the solution space. This differs from the neighbourhood described in [18] for a 2D version of the problem, which also fixes the relative position variables of each pair of boxes not in $\mathcal{B}^{\mathcal{NBH}}$, but does so for all four possible separation directions. By fixing only the largest-separation relation variable, our LNS obtains larger neighbourhoods and simpler models.

6 Evaluation

We have evaluated the practicality of our system by executing it as an 8-thread process on an Intel(R) Core(TM) i7-4771 CPU @ 3.90GHz on two benchmarks. The first one extends the default benchmark of [3], which models the *acid gas removal-1100* unit of an existing plant, by adding maintenance access zone boxes and pipe t-junctions, yielding 39 boxes and 47 pipes. The second benchmark is new and models the combined *dehydration-1300* unit, *mercury removal-1500* unit and *propane circuit of the liquefaction-1400* unit of the same plant. We believe this is the largest plant layout benchmark ever considered in the literature.

Its *container cuboid* is sized $250 \times 100 \times 40$m length by width by height, discretized by 200mm, yielding $1250 \times 500 \times 200$ position points along axes x, y, z, respectively. It has 85 pipes in $\mathcal{P}$, with diameters D_p between 50 and 1400 millimetres, and 76 boxes in $\mathcal{B}$ including: 12 columns and vessels, with heights between 1.5 and 26 meters; 2 heat exchanger groups and 8 individual heat exchangers including 4 fin-fan blocks (the groups have size 22×4×6m, while the sizes of the individual ones range from 6.5×1.5×1.5m to 173×15×5m); 1 source and 9 sink points connecting the equipment to the outside; 2 pipe racks of size 75×15.5×18m and 186×15.5×18m, where levels at heights 3, 6, 9, and 12m provide support without cost; a pump group of size 4×1×4.5 and two compressors of sizes 9×3×3 and 7×5.5×5.5m; 3 general equipment of sizes 7.5×5.5×5.5, 12×10×39 and 25×27×13m; 2 small mixers of size 0.7×0.6×0.6m each; 7 strainers of sizes from 1×0.5×0.5 to 5.5×1.5×1.5m; 17 pipe t-junctions; and 10 maintenance access zones (4 truck, 4 landing and 2 extract zones).

The use of MiniZinc allowed us to try several solvers, including two state-of-the-art MIP solvers (Gurobi 7.5.2 [7] and IBM ILOG CPLEX 12.8 [8]) and the two CP solvers that gave the best performance for phase two in [3] (Chuffed [5] and Gecode 6.0 [15]). For MIP solvers we *warm-start* the solver on each neighbourhood, i.e., the last solution is not "destroyed" [13] by demanding a strictly better objective value, but provided as a solution hint to the solver [7, 8]. This considerably sped-up MIP by allowing the efficient solving of much larger neigh-

Table 1. Unit 1100 with 20 LNS iterations and Unit 1300+ with 50 LNS iterations

		First Solution		Sequential LNS						Random LNS					
				Restart			End (after LNS)			Restart			End (after LNS)		
	Solver	Time (sec)	Total Cost (10^3)	Time limit (sec)	Obj (UB) (10^3)	Gap (%)	End Time (sec)	Obj (UB) (10^3)	Gap (%)	Time limit (sec)	Obj (UB) (10^3)	Gap (%)	End Time (sec)	Obj (UB) (10^3)	Gap (%)
Unit1100	CPLEX	13	751	30	728	24.90	186	561	2.46	30	728	24.90	220	618	11.54
				60	673	18.75	260	556	1.60	60	673	18.75	188	558	1.97
				120	654	16.42	294	556	1.66	120	654	16.42	275	555	1.53
				180	572	4.46	303	558	2.07	180	630	13.17	370	557	1.81
	GUROBI	18	660	30	585	6.48	146	550	0.53	30	585	6.48	122	555	1.53
				60	567	3.53	173	550	0.53	60	567	3.53	152	555	1.53
				120	567	3.53	234	550	0.53	120	567	3.53	212	555	1.53
				180	567	3.51	295	550	0.53	180	567	3.51	276	555	1.53
Unit1300+	CPLEX	146	2721	60	2698	47.23	3818	1597	10.87	60	2525	43.60	3901	1658	14.11
				300	2336	39.06	3967	1657	14.10	300	2277	37.48	4152	1652	13.84
				600	2283	37.63	4348	1654	13.93	600	2229	36.13	4446	1613	11.74
	GUROBI	611	2480	60	2105	32.37	4174	1614	11.79	60	2164	34.20	4273	1553	8.35
				300	1980	28.11	4453	1552	8.24	300	2038	30.14	4492	1555	8.47
				600	1785	20.22	4739	1600	11.02	600	1849	23.01	4888	1609	11.53

bourhoods than in [18]. Note that only the variable values are provided for a warm start.

Table 1 provides the results for our two benchmarks (denoted as Unit1100 and Unit1300+) using the MIP solvers, with parameters L=15 and U=20 for building the neighbourhoods. For each solver and type of neighbourhood, it shows the time in getting the first solution and its associated objective value ($\times 10^3$); the objective value and associated gap of the solution found by the warm-started solver with the given timeout (30, 60 and 120 seconds for Unit1100, and 60, 300 and 600 for Unit1300+), and the objective value and associated gap of the solution found after performing all LNS iterations (20 for Unit1100, 50 for Unit1300+). Note that the gap shown is computed using the best lower bound found in any run. For Unit1100 this is the optimal value, 546797, which is found by Gurobi in 802 seconds. For Unit1300+ it is 1440125.44 after two days of computation with 8 threads, and might still be suboptimal.

In addition, Figures 3 and 4 show information regarding the value of the objective function and the time (also in seconds) at which the associated solution was found for each of the 20 and 50 LNS iterations summarised in the above Table (all starting from the initial solution found by each solver). The rightmost figures for each solver also show the solutions found by that solver without LNS in the same time-frame. As the figures show, Gurobi consistently performs better than CPLEX for our model, and seems to perform most efficiently when given a short time to improve the initial feasible solution to build the seed. Its combination with LNS allows us to provide Woodside engineers with high quality solutions in under 2 minutes for Unit1100 and under 30 minutes for Unit1300+. This is quite pleasing as it is well within the expectations of Woodside, not only for obtaining a first solution, but also for performing interactive re-optimisations. Still, we would like to reduce further the time taken for the second benchmark.

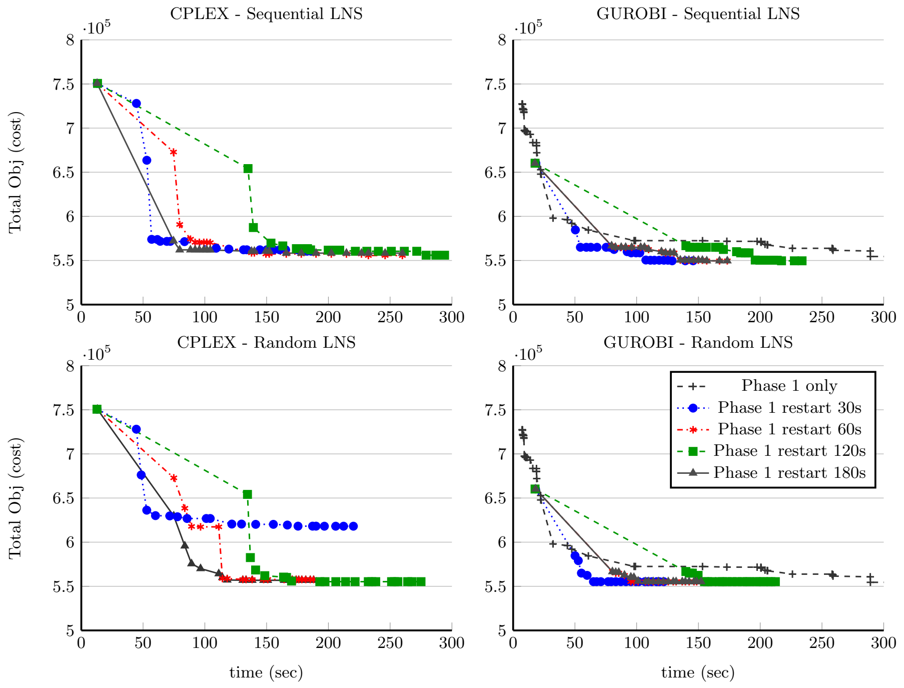


Fig. 3. Intermediate MIP solutions for Unit1100 with and without LNS

Thus, we plan to explore the use of other neighbourhoods, in combination with hierarchical approaches to decompose big plants.

The experiments with CP solvers were not as successful. For Chuffed we were not able to get a first solution in 1 hour, even for the smaller benchmark and trying with a variety of searches (free, model, alternating, etc.). Thus, we seeded it with a solution from Gurobi. Then, the best results were obtained when warm-started with an upper bound on the objective, allowed to alternate between free/user-defined search (-f solving option), and the neighbourhood size was reduced to 5–15. Even then, after 50 iterations it was only able to return an objective of 574247 (and it took 4739 seconds). For Unit1300+ it never returned a solution better than the seed. For Gecode we were again not able to find a first solution for any of the two benchmarks, and seeding it with Gurobi (and setting an upper bound for the objective function) did not produce any improvement, perhaps due to its reliance on the search specified by the model.

7 Conclusions

We have presented the most realistic model ever described in the literature to solve phase one of the plant-layout problem, which positions the equipment ensuring it satisfies directional safety distances, equipment alignment, and various

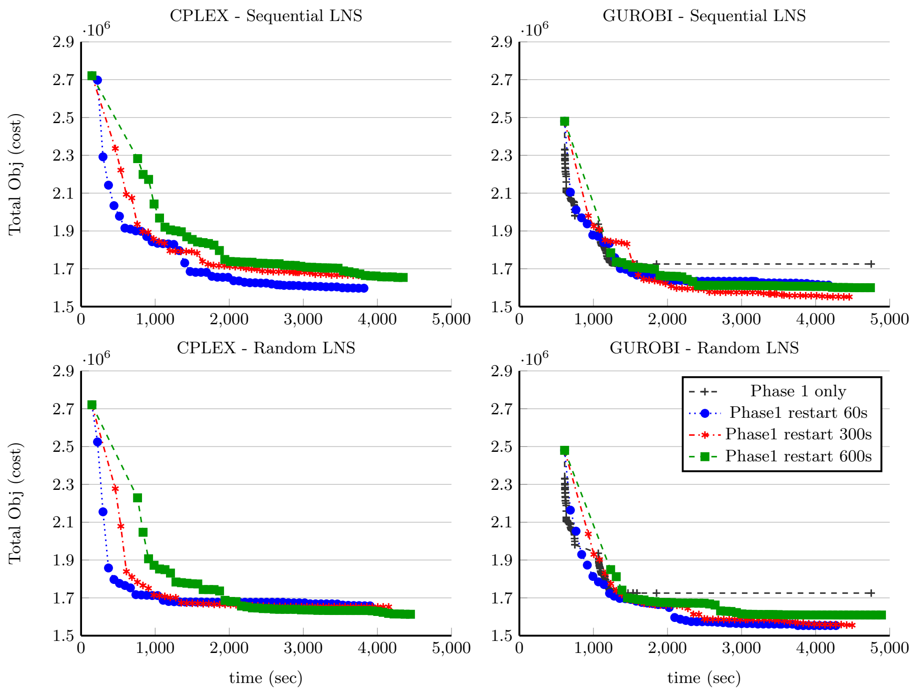


Fig. 4. Intermediate MIP solutions for Unit1300+ with and without LNS

types of (rigid, rotatable and multi-zone) maintenance access constraints in such a way as to minimise the piping and support costs of all equipment and the overall footprint of the plant. Making the model sufficiently realistic to satisfy industry standards has been very challenging and considerably increased the search space and, thus, the time taken by the solvers to find a high-quality solution. Thus, we also developed and implemented an LNS framework that can explore larger neighbourhoods than any previous approach for this problem, thanks to the use of complete solvers able to explore the neighbourhoods efficiently. Our experimental results show that the combination of MIP solvers with LNS provides Woodside engineers with high quality solutions in under 2 and 30 minutes for our two benchmarks, respectively. The use of a constraint programming modelling language (MiniZinc) was critical to be able to modify the model as often as required, and execute it with the most efficient solver for that model.

Acknowledgements. Funded by Woodside Energy Ltd and the Australian Research Council grant DP180100151. We thank our Woodside collaborators, particularly Solomon Faka and Michelle Frayne, for the many useful discussions.

References

1. AMEC Paragon launches optimized FEED design process. *Zeus Technology Mag-*

azine, 4(2):1–3, February 2009.

2. N. Beldiceanu, M. Carlsson, S. Demassey, and T. Petit. Global constraint catalogue: Past, present and future. *Constraints*, 12(1):21–62, 2007.
3. G. Belov, T. Czauderna, A. Dzaferovic, M. Garcia de la Banda, M. Wybrow, and M. Wallace. An optimization model for 3d pipe routing with flexibility constraints. In *International Conference on Principles and Practice of Constraint Programming*. Springer, 2017.
4. G. Belov, P. J. Stuckey, G. Tack, and M. Wallace. Improved linearization of constraint programming models. In M. Rueher, editor, *Principles and Practice of Constraint Programming: 22nd International Conference, CP 2016, Proceedings*, pages 49–65. Springer International Publishing, 2016.
5. G. G. Chu. *Improving combinatorial optimization.* PhD thesis, 2011.
6. R. Guirardello and R. E. Swaney. Optimization of process plant layout with pipe routing. *Computers & Chemical Engineering*, 30(1):99–114, 2005.
7. Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual Version 7.0.* Houston, Texas: Gurobi Optimization, 2016.
8. IBM. *IBM ILOG CPLEX Optimization Studio. CPLEX User's Manual.* 2017.
9. Y. T. Kar and G. L. Shi. A hierarchical approach to the facility layout problem. *International Journal of Production Research*, 29(1):165–184, 1991.
10. J.C. Mecklenburgh. *Process Plant Layout.* Halsted Press; John Wiley & Sons, New York, 1985.
11. N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack. MiniZinc: Towards a standard CP modelling language. In C. Bessiere, editor, *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming*, volume 4741 of *LNCS*, pages 529–543. Springer-Verlag, 2007.
12. M. S. Peters and K. D. Timmerhaus. *Plant design and economics for chemical engineers.* McGraw-Hill Book Company, New York, 5th edition, 2004.
13. D. Pisinger and M. M. Sigurd. Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS Journal on Computing*, 19(1):36–51, 2007.
14. A. Sakti, L. Zeidner, T. Hadzic, B. St. Rock, and G. Quartarone. Constraint programming approach for spatial packaging problem. In C.-G. Quimper, editor, *Integration of AI and OR Techniques in Constraint Programming: 13th International Conference, CPAIOR 2016. Proceedings*, pages 319–328. Springer, 2016.
15. C. Schulte, G. Tack, and M. Z. Lagerkvist. Modeling and programming with Gecode, 2017. www.gecode.org.
16. P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming*, pages 417–431. Springer, 1998.
17. G. Xu and L. G. Papageorgiou. A construction-based approach to process plant layout using mixed-integer optimization. *Industrial & Engineering Chemistry Research*, 46(1):351–358, 2007.
18. G. Xu and L. G. Papageorgiou. Process plant layout using an improvement-type algorithm. *Chemical Engineering Research and Design*, 87(6):780–788, 2009.