Journal of Manufacturing Systems 50 (2019) 180–192

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

An effective integer program for a general assembly line balancing problem with parallel workers and additional assignment restrictions

Bryan W. Pearce, Kavit Antani, Laine Mears, Kilian Funk, Maria E. Mayorga, Mary E. Kurz*

Clemson University, United States

ARTICLE INFO

Keywords:
General assembly line balancing
Heuristics
integer programming models

ABSTRACT

The scope of the assembly line balancing problem in research is clear, with well-defined sets of assumptions, parameters, and objective functions. In application, these borders are frequently transgressed. Many of these deviations are internal to the assembly line balancing problem itself, arising from any of the physical or technological features in modern assembly lines. Other issues are founded in the tight coupling of assembly line balancing with external production planning and management problems, as assembly lines are at the intersection of multiple related problems in job sequencing, part flow logistics, worker safety, and quality. General assembly line balancing is devoted to studying the solution techniques necessary to model these applied line balancing problems.

This article presents a complex line balancing problem based on the real production environment of our industrial partner, featuring several extensions for task-to-task relationships, station characteristics limiting assignment, and parallel worker zoning interactions. A heuristic, combining rank-position-weighting, last-fit-improvement and iterative blocking scheme, and an integer program that can manage multiple constraint types simultaneously, are developed. An experiment is conducted testing each of these new solution methods upon a battery of testbed problems, measuring solution quality, runtime, and achievement of feasibility. Results indicate that the integer programming model provides a viable solution method for those industries with access to commercial solvers.

1. Introduction

The traditional form of an assembly line, as described by Scholl [1], is a production system consisting of a configuration of consecutive workstations, typically using some material handling equipment to transport workpieces down the line at a constant rate. The total work to be performed along the assembly line is subdivided into the smallest indivisible elements of work, called tasks, and each task i possesses an associated task time (t_i). Tasks are related to one another by precedence attributes, *i.e.* some tasks must be finished before others can begin, usually due to the physical architecture of the workpiece. Stations are spaced along the line such that there is one workpiece present at each station, and all stations are allotted a fixed cycle time (C) to execute tasks before the conveyor moves the workpiece to the next station.

The assembly line balancing problem (ALB) is a production planning problem concerned with allocating tasks to the stations on the assembly line, first proposed and formulated as a mathematical programming problem in [2]. A solution to the ALB is an assignment of tasks to stations. One common objective is to minimize the number of stations given a fixed cycle time; this optimization problem is NP-hard [3].

ALB appeared in the literature in the 1950s. By the early 1970s, algorithmic ALB methods had proliferated, yet a survey at that time found that only approximately 5% of companies were using published methods to solve their internal ALB problems [4]. Many articles attest to the continuing prominence of intuitive methods over algorithmic ones developed by the research community, covering all decades of the intervening time period [5–8]. A field book published as recently as 2012 [9] makes no mention of algorithmic methods at all, instead recommending a manual approach, in consultation with a process expert to ensure the balance is feasible. The simplest explanation for the lack of algorithms implemented in industry may be the fact that finding a feasible solution to an ALB can usually be accomplished by hand. The manual solution will perhaps not be optimal, but might at least be good enough to seem acceptable to management. Companies with expertise in product design, as opposed to assembly system design, may not have the information infrastructure to support computational methods of line balancing (perhaps precedence data is unavailable, for example).

* Corresponding author.
E-mail address: mkurz@clemson.edu (M.E. Kurz).

https://doi.org/10.1016/j.jmsy.2018.12.011
Received 9 November 2018; Accepted 24 December 2018
Available online 09 January 2019

Additionally, there are certain normal translational difficulties for any new theoretical work. Industry adoption requires potential adopters to learn that the theoretical methods exist, overcome organizational inertia resisting change, and, of course, financial investment to implement the change.

The real-world ALB problem may possess features that either by themselves, or in conjunction with one another, are not modeled by any published solution procedure. No published contribution offers ALB modeling methods with the constraint detail necessary to capture operational dynamics at a facility manufacturing complex products in a mixed-model environment. The use of ALB solution methods with insufficient constraint modeling renders any generated solution vulnerable to infeasibility. A practitioner might develop new methods as needed, with appropriate background and skill. Given the deliverable-oriented nature of many process engineering job duties, however, it is perhaps uncommon that such research would be undertaken.

In this paper, we describe related literature and the specific problem environment of interest in Sections 2 and 3. We propose the MRPW-LFI-WZBlock Heuristic in Section 4, followed by an integer programming formulation in Section 5. In Section 6, we present the test bed of problems, based on industry problem sets, upon which the heuristic and integer program are evaluated. Section 6 also contains the discussion of the results. Section 7 concludes the paper.

2. Related literature

The classification and terminology suggestions from the surveys of Refs. [1,10,11,12] have been herein adopted, with any conflict favoring the more recent publication. We focus this review on the features most relevant to our industrial setting. Salveson's initial formulation is known as the simple assembly line balancing problem (sALB) [10,11], as it features a number of simplifying assumptions:

1 Mass-production of one homogenous product.
2 All tasks are processed in a predetermined mode (no processing alternatives exist).
3 Paced line with a fixed common cycle time according to a desired output quantity.
4 The line is considered to be serial with no feeder lines or parallel elements.
5 The processing sequence of tasks is subject to precedence restrictions.
6 Deterministic (and integral) task times.
7 No assignment restrictions of tasks besides precedence constraints.
8 A task cannot be split among two or more stations.
9 All stations are equally equipped with respect to machines and workers.

Deviations from the above assumptions are classified as general ALB (gALB) problems. Regardless of the deviation, almost all gALB problems still require the following minimal set of constraints to be satisfied:

- All tasks are assigned to some station, and the workpiece is finished upon exiting the final station.
- All precedence relationships must be satisfied.
- The sum of task times at each station cannot exceed the cycle time.

In addition to precedence constraints, many real-world problems exhibit tooling, zoning, worker skill, and other characteristics that restrict the assignment of tasks to stations and/or the relative assignment of tasks to one another. Tooling availability and worker skill have been addressed in a branch and bound model [13]. High complexity tasks are grouped together such that high skill operators can be assigned to those stations. A hybrid adaptive search and genetic algorithm has examined task sets that are ineligible for assignment within the same station [14]. Additionally, a hybrid heuristic/genetic algorithm approach has been used to minimize a composite fatigue score to address the effects of physical fatigue on workers from tasks that vary in difficulty in a system with a given number of stations and cycle time [15].

Assembly lines with parallel workers allow multiple workers per station and are appropriate for workpieces large enough to permit several workers with simultaneous access, *e.g.* larger home appliances, cars or airplanes. Allowing multiple workers per station may reduce the number of stations required, resulting in corresponding improvements to factory floor space utilization and facility capital construction costs. Consolidation of workers into the same station as one another may also allow sharing of fixed tooling resources between them, reducing capital costs. Material handling costs may also be reduced, as there are fewer destinations to support with part delivery, spread across a smaller footprint. Lastly, multiple worker lines may realize superior line balancing solutions due to reduced worker movement around the workpiece, as each worker can be assigned tasks that only appear in a specific zone [16,17].

The first parallel ALB considered lines in which there are two distinct sides to the assembly line, and proposed a priority heuristic to generate a solution [16]. Tasks are classified into the sets {left, right, either} corresponding to the side of the line to which they may be assigned. A genetic algorithm methodology was later introduced for this gALB problem [18]. An important consideration for simultaneous work is that precedence related tasks might be assigned to separate sides of the line, resulting in one side waiting for the other to finish a task in order to begin work on the next one. Two supplementary objective function were offered to address this issue: 1) work relatedness, which promotes tasks that have an immediate precedence relationship to be assigned to the same station, and 2) work slackness, which promotes those tasks to have a large time gap between them if they cannot be assigned to the same station [17]. A stochastic single-pass prioritization heuristic was created for a two-sided assembly line that manufactures appliances [19]. An ant-colony metaheuristic was proposed for a two-sided assembly line that produces domestic products [20], which was extended to include secondary vertical and horizontal balancing objectives [21]. Previous two-sided ALB models were extended and addressed by a multi-objective particle swarm optimization approach that incorporates work relatedness, utilization, and vertical smoothing [22]. Four work zones and additional constraints to require certain sets of tasks be assigned to the same station were considered and addressed by a hybrid DP and tabu search algorithm to vertically smooth workloads over a given number of stations [23].

Solution methods for ALB problems are often based on the Ranked Positional Weight (RPW) heuristic [24]. RPW is designed for fixed cycle time version of SALB, and attempts to pack stations with tasks by assigning them one at a time starting at the beginning of the assembly line. Each task is sorted by a weight that is equal to its task time plus the task times of all of its successor tasks, which ensures that no task will be before one of its predecessors in the sorted list. Each task, considered in the resulting order, is placed at the earliest station that meets both constraint criteria: 1) sufficient time capacity exists at the station to place the task, and 2) no predecessor of the task is at a later station. The Inverse Positional Weight heuristic is identical to RPW except that tasks are weighted as the sum of their task time and all predecessor task times. The assignment sequence begins at the final station instead of the first [24]. The Related Activity heuristic scores tasks in the same way as the Inverse Positional Weight heuristic but restricts task selection to the task with the largest score that is smaller than the cycle time [25]. The selected task is assigned to a new worker, along with all predecessor tasks. The algorithm then recalculates scores for all unassigned tasks, and repeats the assignment process. Upon completion the set of workers are then sequenced according to precedence validity.

Tonge [26,27], developed a heuristic that groups sets of tasks that are directly connected in the precedence graph and treats these groups as single tasks. The resulting groups are approximately equal in aggregate time. A valid assignment is sought using the much-reduced task

set size. If an assignment is found, the algorithm then seeks to improve the solution by smoothing tasks from one station into an adjacent one, while preserving precedence and cycle time constraint satisfaction. If no solution can be found with a given task grouping, then the groups are broken into smaller subsets and the process repeats. Pinto et al. also utilized the network inspired procedure, in which tasks are grouped together via adjacencies on the precedence graph. These super-nodes are then sequenced and fitted into stations to evaluate the resulting balance. The procedure is iterated with differing heuristic rules for constructing the super-nodes [28].

Baybars developed a set of preprocessing steps oriented to reducing problem complexity [10,11]. The input data is analyzed for complexity saving opportunities, such as decomposing the original problem into smaller sub-problems, or detection of implicit assignment constraints. After preprocessing a heuristic procedure assigns tasks one at a time beginning at the end of the assembly line. The task prioritization metric considers the subset of tasks with no currently unassigned successors, and then chooses the one with the highest number of predecessors. The chosen task is then assigned to the latest possible station.

Tonge utilized a suite of prioritization metrics coupled with a one-at-a-time task assignment approach [29]. After each task assignment, a randomly selected heuristic is used to select the next task. Tonge reported a competitive result from executing multiple runs of this heuristic, though it is likely this performance was due to the wider solution space search afforded by the element of randomness in each run.

The COMSOAL algorithm [30] expanded upon the work of [29]. On the first pass the selection probabilities are uniform between all tasks that have no unassigned predecessors, though these probabilities may change as the algorithm iterates. COMSOAL can be characterized as a form of learning algorithm, as objective function performance is used as an input between iterations to bias the probabilistic selection steps in future iterations. Nine methods are given for inducing bias into the probabilistic search.

3. Problem environment and additional constraints

The motivating context for this research is automotive engineering. We consider a vehicle to be the representative workpiece. Table 1 provides an overview summary of the features of the problem under consideration in this paper. The classification system of Boysen et al. [31] is noted for each feature in the right-most column.

This research utilizes the Type 1 objective function, to minimize the number of workers, as cycle time is fixed by consumer demand and not subject to the line balancing process. While holding cycle time constant, the number of stations in the solution is a function of the optimization performance.

3.1. Parallel workers and zoning constraints

Zoning constraints are introduced to prevent interference between parallel workers in the same station. The physical space that the vehicle occupies on the conveyor is partitioned into a set W of five *work zones (WZ)*: {V (front), R (right), L (left), H (rear), and I (center)}. Every station will have between zero and five assigned workers, each of which is responsible for a single WZ. Tasks must be assigned to both a station and a WZ in order to ascribe them to a unique worker.

Each task i is encoded with one of nine *Product Zones (PZ)* $L = \{RV,MV,LV,RM,MM, LM,RH,MH,LH\}$ corresponding to location on the vehicle with which the task interacts, divided into a 9-zone grid. Fig. 1 displays the WZ and PZ zoning divisions.

Workers may only be assigned a task if the worker's WZ overlaps with the PZ of the task. Each WZ is initially eligible to cover three or more PZ, as shown in Fig. 2. For example, a worker in the V WZ has the RV, MV, and LV product zones within reach.

While Fig. 2 shows all potential matches between PZ and compatible WZ, some pairings cannot be activated simultaneously. To avoid interference problems between workers, each PZ may only be assigned to one WZ within each station. For example, while the LV PZ may be assigned to either the L or the V WZ, it may not be assigned to both within any given station. All tasks that are located in the LV PZ must be assigned to the same worker, either the L or the V worker.

Fig. 3 illustrates this zoning conflict. The lightning bolt flags 1 and 3 show workers attempting to perform tasks upon the same area of the vehicle.

3.2. Task to task constraints

Adjacency requires the involved tasks to be performed consecutively by the same worker. Consider two tasks: Task A requires collecting a part, and Task B installs that part on the vehicle. In addition, Task A must immediately precede Task B, because the worker's hands are full with the part and no other tasks should intervene. A *same-takt (worker)* constraint requires the involved tasks to be performed by the same worker, but not necessarily consecutively. A *same-station* constraint requires the involved tasks to be performed on the same station, but not necessarily by the same worker. Consider the example of headliner installation. Due to the size of the part, installation requires several workers to hold the part during the install. Each of these workers is assigned a different task, but all tasks must be done in tandem. Collectively these tasks must be assigned to the same station. In contrast, the *not-same-takt* constraints forbid the same worker from performing the related tasks, *e.g.* an inspection task that must be performed by a separate worker from the install task.

3.3. Station constraints

Resource constraints involve stations that possess fixed resources that are necessary to complete certain tasks, *e.g.* robotic lift support, pneumatic tooling. There may be one or more of each of these resources distributed along the line, and any task that interacts with a certain resource is forced to be assigned to a station that possesses the resource.

Fixed resources generally are located on one side of the line or the other, and can only be used upon the side of the vehicle that is facing it. Fixed resource coverage zones (TZ) are defined as shown in Fig. 4. A fixed resource covers the six PZ that are on the same side of the vehicle. The fixed resource may satisfy the resource needs of any task in those PZ that is assigned at the station.

Vehicles may assume one of eight orientations relative to the conveyor, and the orientation may change between stations. For example, the vehicle may be oriented nose-first in one station, then rotate 90

Table 1
gALB Problem Features Considered.

Feature	Description	Class
Parallel workers	Up to five workers may be assigned at each station, each with a non-overlapping work area dynamically determined by the set of tasks assigned	$\beta_3 = pwork^5$
Mixed model	Intermixed sequences of different models are produced on the assembly line	$\alpha_1 = mix$
Grouped tasks	Task groups define tasks that must be performed by the same worker, or in the same station	$\alpha_5 = link$
Stationary resources	Tasks that require fixed resources can only be assigned to stations that possess the resource	$\alpha_5 = fix$
Task exclusion	Some tasks cannot be assigned to certain stations	$\alpha_5 = excl$

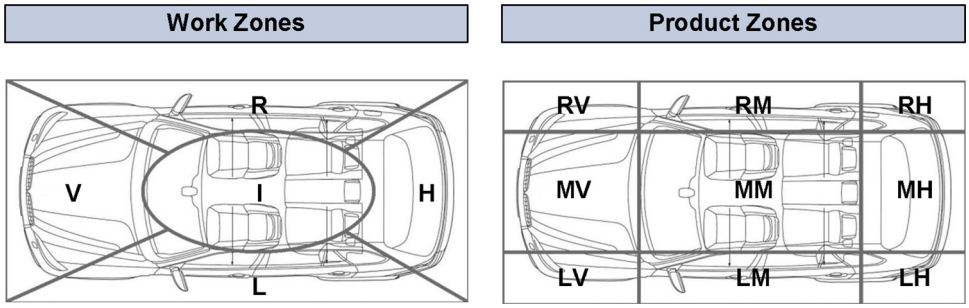

Fig. 1. Work Zones (WZ) and Product Zones (PZ).

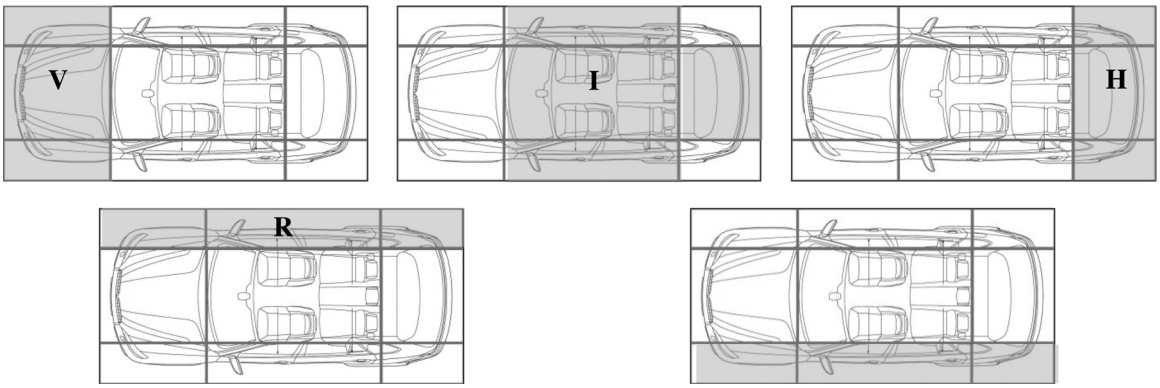

Fig. 2. Product Zones Eligible in each Work Zone.

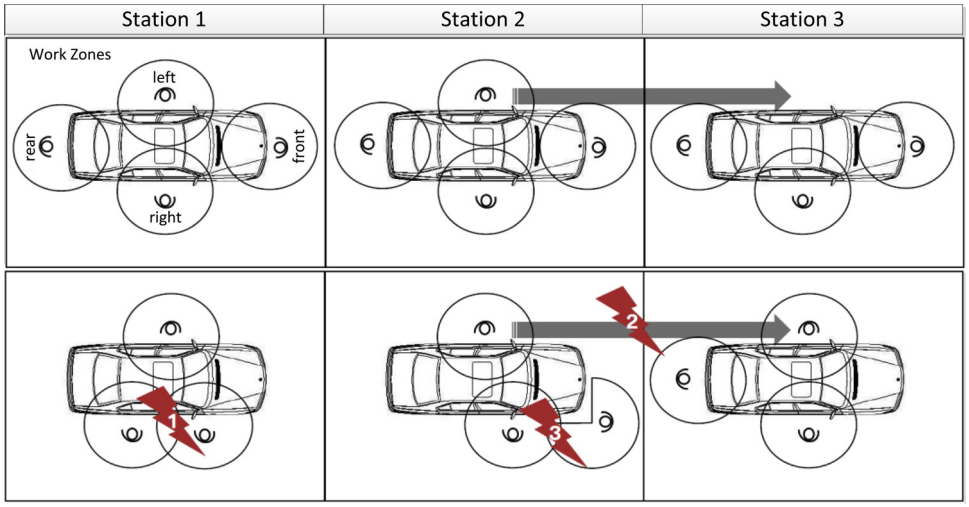

Fig. 3. Zone Conflicts.

degrees such that the vehicle is sideways for the next station. Vehicle orientation strongly affects zoning relationships, dictating which WZ, PZ, and TZ are associated with one another at each station. Appendix B in Supplementary material: IP Preprocessing shows zoning interactions in detail.

Each PZ and WZ at a station may be inaccessible, due to the structural layout of the station. An inaccessible WZ may not have a worker assigned to the zone. Similarly, an inaccessible PZ at a station indicates that tasks of that PZ may not be assigned at that station. Physical obstructions such as pillars, robotic machinery, or the work-piece carrier itself are common causes for inaccessible zone constraints.

The number of workers at any station may be capped at any value 0–5 (there are 5 WZs).

3.4. Notation convention

Table 2 presents the sets over which the parameters and variables in the gALB problem are indexed, and the indexing variable typically used when quantifying over each set. Table 3 presents input parameters for each problem instance.

4. Constructive and improvement heuristics

Concepts from RPW have been extended to manage the features of the production environment in question, including zoning constraints, task groupings, and resource constraints. Several scoring metrics are introduced that prioritize tasks that have difficult satisfaction requirements. The metrics prioritize tasks that require or support successor tasks that require stationary resources on the assembly line.

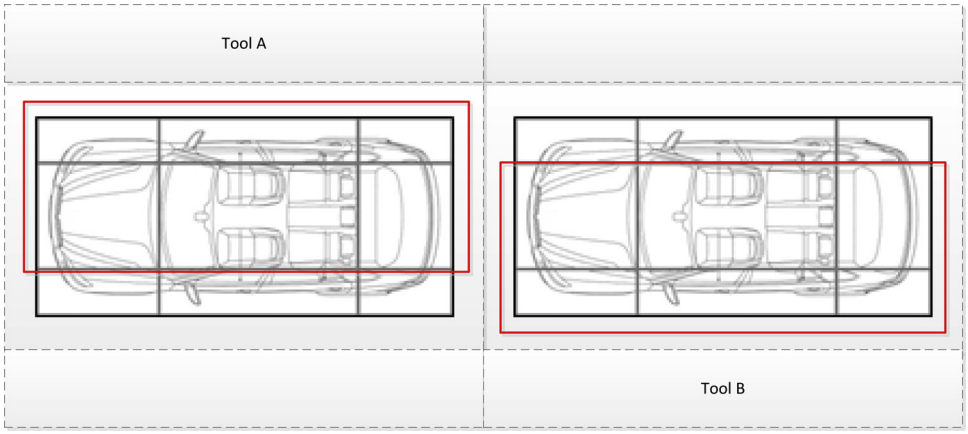

Fig. 4. Fixed Resource Coverage Zones (TZ).

Table 2
Sets.

Symbol	Description	Index
I	set of all tasks $\{1,...,n\}$	i, j
K	set of all stations$\{1,...,\|K\|\}$	k
W	set of all work zones $\{L, R, V, H, I, P\}$	b
L	set of all product zones $\{LV, MV, RV, LM, MM, RM, LH, MH, RH\}$	l
F	set of all fixed resources (such as tools) $\{1,...,\|F\|\}$	f
A	set of assignment restriction types $\{adj, st, ss, nt\}$ (adjacency, same takt, same station, not same takt)	u

Table 3
Problem Input Parameters.

Symbol	Description
C	Cycle time (sec)
t_i	Time of task i (sec)
l_i	Product zone of task i; $l_i \in L$
F_i	Set of fixed resources required by task i; each member is an element of F
p_{ij}	Precedence relation between tasks i and j $p_{ij} = 1$ if i is an immediate predecessor of j; 0 otherwise
a_{ij}^u	Assignment restriction of type u between tasks i and j $a_{ij}^u = 1$ if i and j have relationship $u \in A$; 0 otherwise
F_k^p	Set of fixed resources available at station k in product zone p; each member is an element of F
Q_{if}^u	Fixed resource requirement for task i $Q_{if}^u = 1$ if task i requires fixed resource f; 0 otherwise
s_{kb}^W	Work zone accessibility: $s_{kb}^W = 1$ if WZ b is accessible in station k; 0 otherwise
s_{kl}^P	Product zone accessibility: $s_{lp}^P = 1$ if PZ l is accessible in station k; 0 otherwise
s_k^{max}	Maximum number of workers that may be assigned to station k

Two improvement heuristics are developed in conjunction with the constructive heuristic. The first, LFI, leverages the task prioritization metrics from MRPW to consolidate tasks and remove lightly loaded workers. The second improvement heuristic, termed the WZBlock approach, considers the problem of first selecting work zones then assigning tasks. Two new work zone scoring metrics are developed, oriented towards superior selection of the work zones available for activation.

4.1. Modified ranked positional weight heuristic

The Modified Ranked Positional Weight (MRPW) heuristic, an adaptation of RPW [24], seeks to identify a solution to the gALB problem described in Section 3. Let P_i be the set of all tasks that are predecessors to task i, and Q_i be the set of all tasks that are successors to task i. The ranked positional weight score (r_i) for each task is its own time plus the sum of all successor task times, as shown in (1).

$$r_i = t_i + \sum_{j \in Q_i} t_j \tag{1}$$

A first-fit decreasing (FFD) assignment algorithm assigns tasks one at a time, in descending order of r_i score. The first station that satisfies both precedence and cycle time constraints is assigned the task. The RPW prioritization scheme ensures all predecessors are assigned prior to their successors, as $r_i > r_j$ if i is a predecessor of j, as $j \in Q_i$ and $Q_i \supset Q_j$.

4.1.1. Extension: grouping constraints

Adjacency, same-takt, and same-station relationships require the related tasks to be assigned to the same station and/or worker. All tasks in a same-takt group must also be assigned to the same station, and so may be considered to be within a same-station group. Adjacency groups generalize to same-takt groups using the same logic. Adjacency related tasks, then, may also be considered to be in a same-takt group together as well as being in a same-station group together. The term G_i detailed in (2), represents the group of tasks that are related to task i, after fully extending the domain of each task relationship.

$$j \in G_i \text{ iff } a_{ij}^{adj} = 1,\ a_{ij}^{st} = 1,\ \text{or } a_{ij}^{ss} = 1 \tag{2}$$

See the example in Fig. 5. Tasks 4–6 are in a same-takt group (dashed line oval), and tasks 6 and 8 are in an adjacency group together (solid line oval). Task 8 is implicitly included in the same-takt group. The adjacency group, however, does not expand to include tasks 4 or 5. All four tasks are mutually involved in the same G group.

Eqs. (1) and (3) define a *responsibility set* of tasks that require task i, either directly or indirectly. First define $\bar{Q}_i$ as the set that contains all tasks that succeed task i or any task grouped with task i, but not any of the tasks within the group G_i itself.

$$\bar{Q}_i = \left(\bigcup_{g \in G_i} Q_g \right) \backslash G_i \tag{3}$$

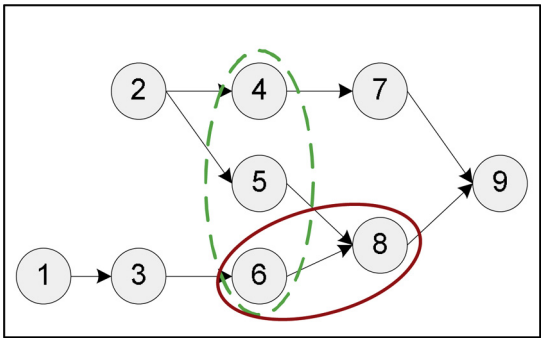

Fig. 5. Overlapping Task Groups.

The responsibility set $\bar{\bar{Q}}_i$ recursively defines the set of all tasks that are dependent on task i, either directly by precedence relationships or indirectly by grouping with tasks that have precedence relationships.

$$\bar{\bar{Q}}_i = \left(\bigcup_{h \in \bar{Q}_i} \bar{\bar{Q}}_h \right) \cup G_i \tag{4}$$

The set G_i is removed from set $\bar{Q}_i$ in (3) to prevent self-referencing recursion in the definition of (4), as would occur in the case of precedence relationships between tasks of the same group. $\bar{\bar{Q}}_i$ contains task i and all tasks in the group G_i, enabling calculation of a modified ranked positional weight score r_i^g that includes all tasks within the responsibility set of i, shown in (5)

$$r_i^g = \sum_{j \in \bar{\bar{Q}}_i} t_j \tag{5}$$

This metric is analogous to combining grouped tasks into single super-tasks, and as a result all tasks $g \in G_i$ will be scored equivalently by r_i^g. Relative scoring to break these ties between in-group tasks can be measured with r_i. Fig. 6 shows the growth of responsibility sets from task groupings.

4.1.2. Extension: resource constraints

Recall that F_i is the set of fixed resources required by task i and $|K|$ is the number of stations available. *Let* F_k^l be the set of resources available at station k in PZ l. Eq. (6) is an *urgency score* that measures the relative importance of fixed resource *res* by the last station to possess *res*. For example, if there are 17 stations and a resource last appears on station 15, then that resource has $Z_{res} = 2$.

$$Z_{res} = |K| - max\{k | \forall\ l,\ res \in F_k^l\} \tag{6}$$

Given two fixed resources res_1 and res_2, if the station number of the final appearance of res_1 is less than the final appearance of res_2 then res_1 will have a higher urgency score, reflecting the fact that tasks that require res_1 have fewer opportunities to assign their predecessors during a first fit decreasing heuristic. To impose prioritization of these predecessors each task i inherits the maximum Z_{res} from their responsibility set $\bar{\bar{Q}}_i$, as shown in (7).

$$r_i^r = max\left\{ Z_{res} \;\middle|\; \forall\ res:\ res \in \bigcup_{j \in \bar{\bar{Q}}_i = F_j} \right\} \tag{7}$$

4.1.3. MRPW algorithm

The MRPW algorithm applies the metrics from (1), (5), and (7) to prioritize tasks according to task grouping, resource, and precedence constraints. These metrics are combined in a hierarchy such that r_i^r dominates, followed by r_i^g, then r_i (Fig. 7).

4.1.4. MRPW remarks

The algorithm calculates r_i, r_i^g, and r_i^r for each task i, by application of (1-(7. In lines 10 and 11 of the MRPW algorithm, the set of unassigned tasks is sorted, first filtering tasks by maximum r_i^r. Ties are broken first by the largest r_i^g, then the largest r_i, and then arbitrarily from the candidates. Next, all tasks that are linked to task i via adjacency, same-takt, or same-station constraints are assigned to a group which will all be assigned to the same WZ or station as appropriate.

Lines 14 and 15 describe finding the station at which to begin the search, by considering precedence. On line 17, three conditions are considered for assigning task i. First, if there is a task j at this station with the same PZ as task i, then the only WZ at this station to which task i can be assigned is the same WZ to which j is assigned. The second condition checks whether the WZ that contains j has sufficient capacity to add task i. The third condition checks whether the resource needs of task i are met at this location. If all of these conditions hold then task i is assigned to this station and WZ.

Lines 20–23 consider all WZ at this station that are not empty (possess at least one task). The motivation here is to attempt to add task i to an existing WZ if possible, rather than open a new WZ. If there exists an already open WZ that can hold task i's PZ, and that WZ has sufficient time capacity, and the resource needs of task i are met at this location, then assign task i to that WZ.

The logic on line 25 considers relaxing the restriction that the WZ be non-empty. If the count of WZ with tasks assigned has not yet hit the s_k^{max} limit at this station, then perhaps a new WZ can be opened to hold task i. The time capacity and resource satisfaction assignment conditions must be met here.

Line 30 increments to the next station, as a feasible assignment at the current station was not found. Line 31 checks whether the new station index exceeds the number of stations given as input. If so, then the new station under consideration will not have any information regarding resource availability. This is considered a failure mode, as the task will be assigned to a station beyond the bounds of the given input data. All known resources are given to the new (dummy) station to ensure that task i can be assigned.

4.2. Last-Fit-Increasing improvement heuristic

The Last-Fit-Increasing (LFI) improvement heuristic is a one-at-a-time task reassignment method that seeks to consolidate tasks from lightly loaded workers. It intends to improve the FFD solution by moving tasks to as late in the assembly line as possible without

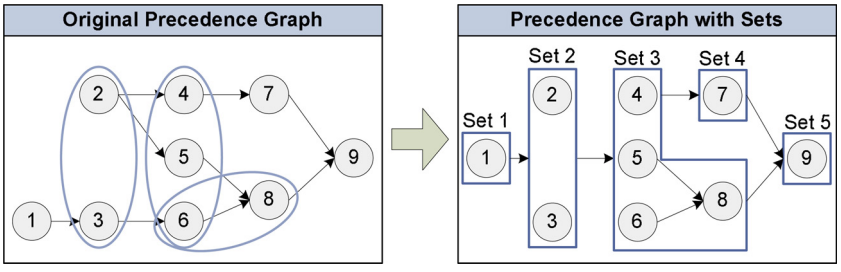

Fig. 6. Group Definition of Responsibility Sets.

```
1   Algorithm ModifiedRankedPositionalWeight
2
3       Let I^free contain unassigned tasks.
4       Set I^free = I
5       Let W contain stations.  Set W = {1}
6       Let S_k contain the tasks assigned to station k
7       Let S_k^b contain the tasks assigned to WZ b in station k
8       Set S_1 = ∅
9       Start: While I^free ≠ ∅
10          Select i = argmax_i{Mr_i^r + r_i^g | i ∈ I^free}
11          In case of tie, i = argmax_i{r_i | i ∈ Set of tied tasks}          // a task is selected
12          Collect group G_i
13          Remove tasks in G_i from I^free            // all tasks in its group will be assigned
14          If P_i = ∅, ∀i ∈ G_i, k = 1
15          Else k = argmax_m{∀j ∈ P_i: j ∈ S_m: i ∈ G_i}     // attempt to put these tasks at station k
16          While G_i unassigned
17              If ∃j: l_j = l_i, j ∈ S_k^b AND t(S_k^b) + Σ_{j∈G_i} t_j ≤ C AND F_i ∈ F_k^l
18                  S_k^b = S_k^b ∪ G_i, (assign group to station k, WZ b)
19                  GoTo Start
20              ∀b: S_k^b ≠ ∅
21                  For each b that is compatible with l_i (PZ map eligible)
22                      If t(S_k^b) + Σ_{j∈G_i} t_j ≤ C AND F_i ∈ F_k^l
23                          S_k^b = S_k^b ∪ G_i (assign group to S_k^b)
24                          GoTo Start
25              If |∀b: S_k^b ≠ ∅| ≤ s_k^max, ∀b:
26                  For each b that is compatible with l_i (PZ map eligible)
27                      If t(S_k^b) + Σ_{j∈G_i} t_j ≤ C AND F_i ∈ F_k^l
28                          S_k^b = S_k^b ∪ G_i (assign group to S_k^b)
29                          GoTo Start
30              k = k + 1
31              If k > |K|                                          // This is a failure mode
32                  |K| = |K| + 1
33                  S_k^b = ∅∀b ∈ B (empty station)
34                  F_k^b = F (give all known fixed resource)
```

Fig. 7. MRPW Algorithm.

```
1   Algorithm LFI_Improve
2
3       Let W contain stations.
4       Let S_k contain the tasks assigned to station k.
5
6       Set I^free = I
7       While I^free ≠ ∅
8           Select i = argmin_i{Mr_i^r + r_i^g | i ∈ I^free}
9           In case of tie,  i = argmin_i{r_i | i ∈ Set of tied tasks}
10          Collect group G_i
11          Remove tasks in G_i from current assignment and I^free
12          k = argmin_m{∀i ∈ P_j: j ∈ S_m: i ∈ G_i}
13          While G_i unassigned
14              If ∃j: l_j = l_i, j ∈ S_k^b AND t(S_k^b) + Σ_{j∈G_i} t_j ≤ C AND F_i ∈ F_k^l
15                  S_k^b = S_k^b ∪ G_i, (assign group to station k, WZ b)
16                  GoTo Start
17              ∀b: S_k^b ≠ ∅
18                  For each b that is compatible with l_i (PZ map eligible)
19                      If t(S_k^b) + Σ_{j∈G_i} t_j ≤ C AND F_i ∈ F_k^p
20                          S_k^b = S_k^b ∪ G_i (assign group to S_k^b)
21                          GoTo Start
22              k = k - 1
```

Fig. 8. Last Fit Increasing Improvement Heuristic.

disrupting the current allocation of tasks, hopefully resulting in at least one empty WZ. Fig. 8 presents an algorithmic view of this logic. LFI begins with a feasible ALB solution, and utilizes the MRPW task metrics r_i, r_i^g, and r_i^r. In contrast to FFD, however, LFI in lines 8 and 9 considers all tasks in *increasing* order of priority, first selecting the lowest-priority, last-assigned task during MRPW. The small priority score for this task indicates that it is maximally free for assignment anywhere on the assembly line, relative to other tasks, as constraints have not inflated its MRPW metric. This task is a good candidate for pushing as far to the end of the line as possible, so that it might be far out of the way of other tasks with more demanding constraints. The task is moved to the last already-active worker that can feasibly accept it (new workers may not be activated during LFI). Iteration then continues to the task with the next-lowest MRPW priority.

```
1   Algorithm WZBlock
2
3       Variables:
4           WZBlocks, a 2-D array of size |K| × |W|
5
6       If max{Comp_{i,b,k}} ≤ 0, return
7       Save LBinit in LBbest
8
9       WZBlocks(*, *) = False
10      iter = 0
11      While iter < maxIter
12          Reset Comp_{i,b,k}
13          Select WZ b' at station k' that has tasks with the smallest positive Score_{b,k}
14              WZBlocks(k', b') = True
15              Comp_{i,b',k} = False
16                 LBtest = ModifiedRankedPositionalWeight with WZ b at station k blocked
17                 LBtest = LFI_Improve(LBtest)
18                 Update LBbest
19          iter = iter + 1
20      Loop
21
22      Return LBbest
```

Fig. 9. Work Zone Blocking Heuristic.

Consider all the stations in which successors of tasks in G_i are assigned. In line 12 the initial station to begin the search, k, is chosen to be the earliest of those stations. This is the farthest that group G_i might go toward the end of the line, due to precedence. Whichever successor task is found during this search has already moved previously in the course of the LFI, as it has a lower MRPW score. Station k is examined to determine if active workers can feasibly absorb group G_i. If so, the tasks are assigned and the loop proceeds to the next task group. If not, then the previous station will be considered. If no later station is found to which group G_i can move, then the group will simply be reassigned at their originally assigned station.

4.3. Work zone blocking improvement heuristic

Solutions to the gALB problem specified in Section 3 typically activate only a fraction of the WZs available. Solving the problem may be considered in two sequential stages: first, to choose which WZ are activated, then to assign tasks to them. In the spirit of leveraging aggregate task PZ to WZ compatibility patterns, two metrics are introduced that provide insight into the relative quality of activating each WZ.

4.3.1. Work zone metrics

Let $Comp_{i,b,k}$ be an indicator variable on whether task i is compatible with WZ b at station k, based on the following constraints:

1 Fixed resources. If task i requires any fixed resources, then the TZ at WZ's station must provide them.
2 Accessibility. This checks both WZ and PZ accessibility. The PZ of task i must not be blocked at this station, nor may WZ b itself be blocked.
3 Zone overlap. The PZ of task i must be associated with WZ b for possible assignment, as per the mapping provided in Fig. 1.

Moreover, these conditions must be met for all tasks that are grouped with task i via adjacency, same-takt, or same-station linkages, not just for task i itself.

The first WZ metric is a "uniqueness" score, given in (8). The internal term $\sum_{b' \in W, b' \neq b} Comp_{i,b',k}$ counts how many other WZ are compatible with a given task i. This quantity is divided by the total number of other WZ on all stations combined, $|W| \times |K| - |W|$, and subtracted from 1, yielding the fraction of non-compatible WZs for task i. This value is then maximized over the set of all i, subject to i being compatible with WZ b at station k. The final value delivered, $Uniqueness_{b,m}$, is a measure of the maximum degree to which WZ b at station k is needed by any task, normalized on the [0,1] scale. A measure of 0 uniqueness indicates that WZ b at station k is not particularly important to any task, as any task that is compatible with m b is also compatible with every other WZ. On the other hand, a measure of 1 uniqueness indicates that there exists some task for which WZ b at station k is the only possible assignment.

$$Uniqueness_{b,k} = Max_{i \in I}\{1 - \frac{\sum_{b' \in W, b' \neq b} Comp_{i,b,k}}{|W| \times |K| - |W|} | Comp_{i,b,k} = 1\} \tag{8}$$

The second WZ metric is the "flexibility" score, given in (9). $Flexibility_{b,k}$ is the proportion of tasks that may be assigned at WZ b at station k. A measure of 1 flexibility indicates that the WZ is compatible with every task. Zero flexibility indicates that the WZ is compatible with no tasks.

$$Flexibility_{b,k} = \frac{\sum_{i \in I} Comp_{i,b,k}}{n} \tag{9}$$

Both the flexibility and uniqueness metrics provide insight into the relative usefulness and importance of each WZ. Both metrics are normalized to the scale [0,1]. To support the WZBlock heuristic, the two metrics are added together to create a single composite score for each WZ:

$$Scoreb,\ k = Uniqueness_{b,k} + Flexibility_{b,k} \tag{10}$$

4.3.2. WZBlock heuristic algorithm

The WZBlock algorithm proceeds by iteratively blocking work zones from usage, by simulating accessibility constraints additional to any that may be in the original problem data. The approach aspires to identify and forbid the WZs that, if chosen for activation, are most likely to cause infeasibilities or sub-optimality in the objective function. The sum of the flexibility and uniqueness metrics presented in Section 4.3.1 is used to discriminate between WZs. The algorithm shown in Fig. 9 details the procedure.

The variable LBbest retains the best solution found through the course of the algorithm. During each iteration, WZ b at station k with the smallest positive *Flexibility* + *Uniqueness* score is chosen, and WZ k is blocked ($s^W_{kb} = 0$). For redundancy purposes, composite scores of zero are not targeted, as a zero flexibility implies that no task can be assigned to the WZ regardless. The MRPW heuristic is then applied with the targeted WZ blocked from activation, followed by the LFI_Improve heuristic. The forthcoming solution retained if it is an improvement

Table 4
Criteria for Retention of Incumbent Solution.

LBtest	LBbest	ACTION
FEASIBLE	FEASIBLE	Retain higher utilization
FEASIBLE	INFEASIBLE	Retain LBtest
INFEASIBLE	FEASIBLE	Retain LBbest
INFEASIBLE	INFEASIBLE	Retain fewest dummy stations. If tied, retain higher utilization

upon the best-yet-found solution, whereupon the WZBlock algorithm considers the WZ to block for the next iteration. Looping continues until the iteration count exceeds a user-defined maxIter hyper-parameter, or no WZs are identified for potential blockage.

Heuristic solutions might not be feasible. Infeasible solutions contain "dummy" stations with tasks that could not otherwise be assigned to any station. The "update LBbest" command checks whether candidate LBtest is superior to the incumbent LBbest. Feasible solutions are preferred over infeasible ones, using utilization (efficiency) as a metric of goodness. Table 4 presents the four possible scenarios, and corresponding action.

4.4. MRPW-LFI-WZBlock Heuristic

The heuristic proposed includes the three elements previously described and is refered to as the MRPW-LFI-WZBlock Heuristic. WZBlock is used as the outer wrapper heuristic, with maximum iteration hyper-parameter set to 10. Within each iteration, the MRPW constructive heuristic is executed, followed by the LFI improvement heuristic.

The computational complexity of each iteration of the WZBlock heuristic is $O(n\ |K|)$, where n is the number of tasks and $|K|$ is the number of stations. The computational complexity is $O(n^2\ |K|)$ for the MRPW and LFI heuristics.

5. Integer programming model

In this section an integer programming (IP) formulation is presented that models the gALB problem introduced in Section 3. Notation for the sets and input parameters are shown in Section 3.4. As discussed in Appendix B in Supplementary material: IP Preprocessing, many of these input parameters are preprocessed before execution of the IP. Outputs from the preprocessing routine are shown in Table 5. Section 5.1 presents the IP formulation in three parts: decision variables, objective function, and constraints.

5.1. IP formulation

5.1.1. Decision variables

$$x_{ikb} = \begin{cases} 1\ if\ task\ i\ is\ assigned\ to\ station\ k\ and\ WZ\ b \\ 0\ else \end{cases}$$

$$y_{kb} = \begin{cases} 1\ if\ WZ\ b\ at\ station\ k\ is\ active \\ 0\ else \end{cases}$$

5.1.2. Objective

The objective function, z, minimizes the number of workers, as measured by the count of active WZ, modifying the classic objective to conflate WZ (instead of stations) with workers.

$$minimize\ z = \sum_{k \in K} \sum_{b \in W} y_{kb}$$

5.1.3. Constraints

Table 6 presents formulas for all constraints in the IP. The ID column is referenced in subsequent text to provide description for the mechanics of each constraint.Constraint set (C1) enforces each task to be assigned to exactly one worker. Constraint set (C2) ensures the average workload assigned to each active worker cannot exceed the cycle time. Constraint set (C3) ensures no more than s_k^{max} workers may be active at station k. Constraint set (C4) enforces precedence constraints. Constraint set (C5) enforces adjacency and same-takt constraints. Constraint set (C7) enforces same-station constraints. Constraint set (C8) enforces not-same-takt constraints; this is a clique inequality, enforced only for cliques defined by each not-same-takt group. Constraint set (C9) enforces fixed resource constraints. Constraint set (C10) enforces WZ accessibility constraints. Constraint set (C11) enforces PZ accessibility constraints. Constraint set (C12) enforces zoning compatibility. Tasks may only be assigned a WZ at some station if the PZ of the task is compatible with that WZ. See Appendix B in Supplementary material: IP Preprocessing for derivation of the B parameter. Constraint set (C13) enforces zone overlap constraints; this constraint considers all task pairs i and j that share the same PZ, indicated by the preprocessing parameter $w_{ij} = 1$. If i and j are assigned to the same station, and task i is assigned to WZ b, then task j must also be assigned to WZ b. This is accomplished by restricting j from assignment to any other WZ that is not b. This constraint prevents workers from interfering with one another. Without this constraint it would be possible for two workers to simultaneously attempt tasks within the same PZ. Constraint sets (C14) and (C15) ensure all decision variables are binary.

6. Application of solution methodologies

This section describes a computational experiment for the MRPW-LFI-WZBlock Heuristic and the IP formulation. For the IP, performance criteria of interest relate to the computer time requirements necessary to solve the problem, and the scaling of this time as a function of problem size. The primary criteria of interest for the heuristic method is the quality of the solution, as measured by optimality gap.

6.1. Test data sets

The ALB literature provides no testbed data sets that exhibit all constraints desired for this paper. Three sets of test data were collected during the development of these methods, in conjunction with our industrial partner. The three data sets corresponding to assembly line sections are labeled "Band 1", "Band 26", and" Band 30." Table 7 summarizes some properties of each of these initial data sets.

These three initial data sets form the testbed basis of the experiment. In addition, an array of ALB sub-problems are created from the

Table 5
Input Parameters Derived During Preprocessing.

Symbol	Description
Q^{ϵ}_{klf}	Fixed Resource Coverage: $Q^{\epsilon}_{klf} = 1$ if station k has fixed resource f that can be used in PZ l
B_{kbl}	WZ−PZ compatability: $B_{kbl} = 1$ if WZ b can contain PZ l at station k; 0 else
w_{ij}	Indicates that tasks i and j have the same product zone. $w_{ij} = 1\ if\ l_i = l_j$; 0 else

Table 6
IP Constraints.

Constraint Formula	Quantification and Condition	Set
$\sum_{k\in K}\sum_{b\in W} x_{ikb} = 1$	$\forall\, i \in I$	C1
$\sum_{i\in I} t_i x_{ikb} \le C y_{kb}$	$\forall\, k \in K, b \in W$	C2
$\sum_{b\in W} y_{kb} \le s_k^{max}$	$\forall\, k \in K$	C3
$\sum_{k=v+1}^{\lvert K\rvert}\sum_{b\in W} x_{jkb} \le 1 - \sum_{k=1}^{v}\sum_{b\in W} x_{ikb}$	$\forall\, v = 1...\lvert K\rvert - 1,$ if $p_{ij} = 1$	C4
$x_{ikb} = x_{jkb}$	$\forall\, i, j \in I, k \in K, b \in W,$ if $a_{ij}^{adj} = 1$ or $a_{ij}^{st} = 1$	C5
$\sum_{b\in W} x_{ikb} = \sum_{b\in W} x_{jkb}$	$\forall\, i, j \in I, k \in K,$ if $a_{ij}^{ss} = 1$	C7
$x_{ikb} + x_{jkb} \le 1$	$\forall\, k \in K, b \in W,$ for i and j where $a_{ij}^{nt} = 1$	C8
$Q_{ij}^{u}(\sum_{b\in W} x_{ikb}) \le Q_{k,li,f}^{c}$	$\forall\, i \in I, k \in K, f \in F$	C9
$x_{ikb} = 0$	$\forall\, i \in I, k \in K, b \in W,$ if $s_{kb}^{W} = 0$	C10
$\sum_{b\in W} x_{ikb} = 0$	$\forall\, i \in I, k \in K,$ if $s_{kb}^{P} = 0$	C11
$x_{ikb} = 0$	$\forall\, i \in I, k \in K, b \in W,$ if $B_{kbl_i} = 0$	C12
$x_{ikb} + \sum_{b'\in W\setminus b} x_{jkb'} \le 1$	$\forall\, i, j \in I,\ k \in K, b \in W,$ if $w_{ij} = 1$	C13
$x_{ikb} \varepsilon \{0,1\}$	$\forall\, i \in I, k \in K, b \in W$	C14
$y_{kb} \varepsilon \{0,1\}$	$k \in K, b \in W$	C15

Table 7
Test Data Set Properties.

Data Set	# Stations	# Tasks	# Unique FIXED RESOURCES
Band 1	13	396	12
BAND 26	9	317	12
Band 30	10	300	3

initial data sets using the tasks assigned in a provided feasible solution, from every possible partition of three or more adjacent stations. This resulted in 125 datasets with between 58 and 395 total tasks over three to 13 stations. Data sets are available from the authors upon request.

6.2. Experiment execution

The IP was applied to each data instance and executed on the Linux-based Palmetto Cluster at Clemson University. The IP formulation is modeled in AMPL, and run using the Gurobi 5.0 Linux 64 solver. For each problem instance 8 processors and 120gb of RAM are allocated.

MRPW-LFI-WZBlock is applied to each data instance with the hyper-parameter set to 10. This setting is chosen because offline testing has shown limited improvement in objective function for values beyond 10. MRPW-LFI-WZBlock is implemented in VBA, and executed on a 64-bit Windows PC with 2.40 GHz processor and 2GB RAM. The experimental data is summarized in Table 7.

6.2.1. Heuristic feasibility

Some instances of the data were not amenable to solution with MRPW-LFI-WZBlock, and entered into the aforementioned “failure mode”. The instances that did not result in solution from MRPW-LFI-WZBlock are included in Table 8, with a separate row, italicized font, and an average gap of ∞. MRPW-LFI-WZBlock found feasible solutions for 106 out of 125 problem instances. Of the 19 heuristic-infeasible instances, 17 of them are sourced in Band 26 data, and 2 are from Band 1.

6.2.2. IP runtime

The average time to execute the IP model was 3.21 s, aggregated across all datasets; the IP model successfully solved all 125 instances. The maximum time is under 25 s. It is remarkable to have solved a 400-task problem, easily a middle-sized problem by ALB standards, in only 20 s.

Fig. 10 presents AMPL Elapsed Time versus task and station counts, with each band’s datasets collected separately. Fig. 10 shows particularly strong differentiation between bands, and consistency within bands. It appears that there are some characteristics particular to each band which carry strong implications towards the IP runtime.

Table 8
Summary Results (Feasible and Infeasible for MRPW-LFI-WZBlock).

Band	Number of Stations	Number of Instances	Average AMPL Elapsed Time (s)	MRPW-LFI-WZBlock	
				Average Time (s)	Average gap
1	3	11	0.16	24.00	0.30
	4	9	0.32	30.90	0.28
	4	*1*	*0.09*	*24.43*	*∞*
	5	8	0.65	13.74	0.22
	5	*1*	*0.21*	*28.37*	*∞*
	6	8	1.07	7.70	0.26
	7	7	3.68	57.33	0.28
	8	6	5.37	59.55	0.28
	9	5	7.54	60.81	0.23
	10	4	9.67	60.75	0.38
	11	3	11.75	60.19	0.44
	12	2	15.85	48.49	0.56
	13	1	20.44	60.16	0.53
26	3	3	0.21	26.27	0.33
	3	*2*	*0.23*	*29.47*	*∞*
	4	2	0.54	30.59	0.48
	4	*2*	*0.38*	*30.44*	*∞*
	5	1	1.53	10.44	0.57
	5	*3*	*0.77*	*1.66*	*∞*
	6	*4*	*1.23*	*9.21*	*∞*
	7	*3*	*1.99*	*60.37*	*∞*
	8	*2*	*3.18*	*10.36*	*∞*
	9	*1*	*6.30*	*52.63*	*∞*
30	3	8	0.32	1.60	0.00
	4	7	0.87	11.82	0.00
	5	6	1.88	19.04	0.00
	6	5	3.35	31.47	0.00
	7	4	5.02	42.15	0.04
	8	3	9.24	48.40	0.14
	9	2	12.47	54.72	0.13
	10	1	20.62	60.39	0.13

6.2.3. MRPW-LFI-WZBlock optimality gap and time

The optimality gap for instance i is measured by $Gap_i = \frac{z_i - z_{opt,i}}{z_{opt,i}}$, where $z_{opt,i}$ is the optimal value of the objective function, as determined by the IP solution, and z_i is the value of the objective function found by the heuristic. If the heuristic finds an optimal solution, then the gap is zero. The heuristic found 31 optimal solutions of 125, or approximately 25% of problem instances. Of the 31 optimal instances, 29 were from Band 30, two were from band 1 and none were from Band 26. The

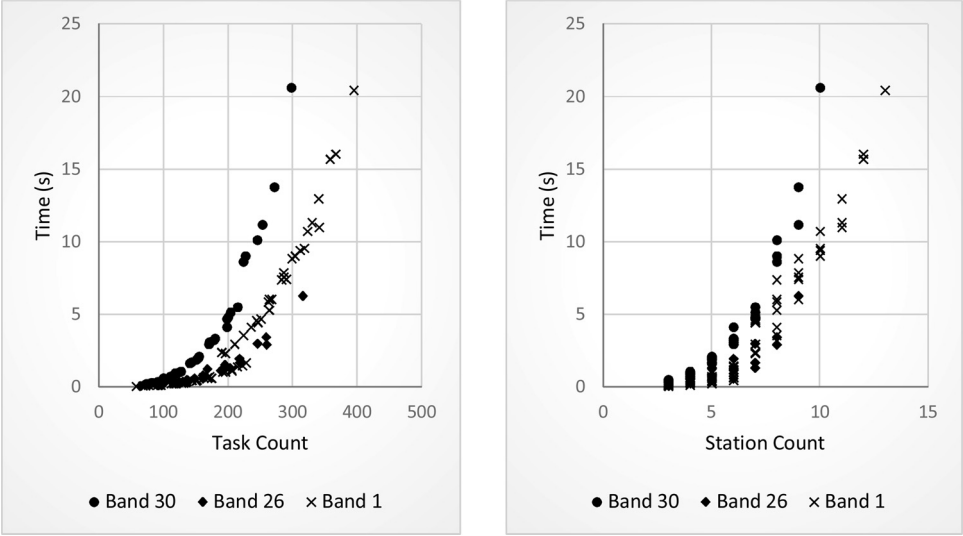

Fig. 10. AMPL Elapsed Time by Band and Task, Station Count.

average gap in the cases in which the optimal solution is not found is 0.204, and in the worst instance is 0.62.

The average time to execute MRPW-LFI-WZBlock is 31.9 s. While this is difficult to compare with the IP running time, due to the platforms on which each is executed, the 10-fold increase is notable.

6.3. Discussion

The runtime plots of the IP suggest accelerating growth in runtime with respect to the size of the problem. Indeed, ALBs are NP-hard, and extremely large problems will certainly be intractable. The instances solved here span up to 400 tasks and 13 stations, what might be considered mid-sized problems in ALB, with runtimes under one minute.

Each band represents an independent production process, with several key differences that might help illuminate band-specific differentiation in performance. All tasks in Band 1 belong to one of the four corner PZs: {LV, RV, LH, RH}, and in most stations workers may only be assigned to the L or R work zones. Only two stations support three parallel workers (the rest cannot have more than two), limiting the potential for overlapping work zones. There are many fixed resources in Band 1, but for most tasks that require fixed resources there is only one WZ which can satisfy both fixed resource and zoning needs, simplifying the decision problem by forcing task assignment. Band 26 is relatively complex, with the full complement of up to 5 parallel workers permitted at many stations. Tasks are located in every PZ. Fixed resources are common, though many are duplicated across two or more stations, permitting tasks requiring fixed resources to be assigned in one of several WZs. Band 30 is a single-sided assembly line, with only one worker permitted per station. Every task is located in the same PZ, and fixed resources are sparse on the line. Task grouping is relatively common in Band 30, but otherwise this band is easily the simplest of the three with respect to constraint complexity.

Fig. 10 shows Band 30 instances requiring the most IP time to solve, Band 26 the least, and Band 1 in the middle. For any given problem size (in terms of either task or station count), Band 30 instances required approximately 4x runtime relative to Band 26 instances, and approximately 2x runtime relative to Band 1 instances. It appears that additional constraints lower IP runtime in general, as a highly constrained instance has a smaller feasible region within the solution space.

7. Summary and conclusions

The MRPW-LFI-WZBlock Heuristic heuristic incorporates three methods. The first, MRPW, introduces a prioritization scheme driven by measuring constraint satisfaction scarcity. Responsibility sets are introduced to encapsulate task-to-task precedence and assignment linkage constraints. Urgency score is introduced to measure assignment limitations due to resource constraints. Tasks are weighted by a composite prioritization score based on these new metrics, and assigned according to a first-fit-decreasing single-pass heuristic. The MRPW heuristic is oriented toward creation of feasible solutions, with efficiency being a secondary consideration. The second component, the Last Fit Increasing improvement heuristic leverages the task prioritization rankings of MRPW, and consolidates flexible tasks into otherwise lightly-packed workers. The goal of the LFI improvement heuristic is to improve the efficiency of a feasible solution. The Work Zone Blocking heuristic focuses on the first work zone selection subproblem of the bifurcated ALB problem. The purpose of this approach is to address zoning difficulties encountered in the MRPW heuristic. Two new metrics are introduced to support the heuristic, measuring work zone flexibility and uniqueness. The metrics are developed in consideration of each WZ's offerings in terms of satisfying task needs with respect to zoning, fixed resource, and accessibility constraints.

An IP is developed to manage the zoning and worker parallelization aspects of the problem. Preprocessing transformations render several complex facets of the problem into representations amenable for a tight IP formulation.

Each solution methodology is applied to a testbed of 125 instances derived from real ALB data collected in conjunction with our industrial partner. The IP is benchmarked primarily according to the runtime required relative to the size of the instance, to which it performs surprisingly well, needing only 22 s at most to solve an instance. The IP solution is also used to benchmark heuristic performance. The heuristics were able to find feasible solutions for 83.2% of problem instances. The heuristics averaged an optimality gap of approximately 20%, and found the optimal solution for 25% of the instances. Due to superior performance and adaptability, the IP is recommended for industrial application.

7.1. Implications for future work and industry implementation

A manufacturer with operations encapsulated by the gALB characteristics for this problem can implement these methods. The algorithmic approach is suitable for embedding within a commercial line balancing visualization software tool. Direct use cases include support for initial (product launch) line balancing, and periodic rebalancing to respond to forecasted demand changes. Secondary use cases include exploration of input parameter variation, *e.g.* cycle time changes, or

relocation of fixed equipment.

Industry application of ALB methods commonly encounter difficulties in extending existing methods to account for gALB features specific to the problem environment. For the gALB environment considered by the methods here, no existing ALB methods were suitable for immediate application, largely due to the unique zoning features. During the course of this research, the heuristic methods were created first, and the IP formulation later. More than a year was spent working with our industry partner, both to collect data and to understand the various constraints that appear in the problem. Some constraint types, *e.g.* not-same-takt constraints, are especially rare on the assembly lines under study, and were not discovered until late in the process. The ALB methods detailed in this work are certainly extensible for application to problem domains outside our industrial partner's, for which the methods were specifically designed. Issues related to industrial application of the IP and heuristic methods are discussed separately in Sections 7.1.1 and 7.1.2, respectively.

7.1.1. IP formulation result

The IP performed exceptionally well for all problem instances in the experiment. Assuming availability of a solver such as Gurobi, it is the recommended solution to any industry customer with an applicable and comparably-sized ALB problem. It is difficult to speculate on IP runtime performance for problems larger than in the experiment, as runtime will certainly experience combinatorial growth rates at some size. Perhaps problems up to an order of magnitude of the largest instances in this experiment (similar in size to the largest ALB problems considered in any literature) would still find acceptable runtimes.

The IP is particularly well-suited for constraint extensions that involve task-to-task or task-to-station assignment compulsion or forbiddance. Several constraints of this variety already exist within the current gALB problem, implemented with relatively clear, direct, and tight IP constraints. Presumably, extending the IP for another gALB environment by adding more constraints of this type would be relatively simple. Indeed, during development of this IP the not-same-takt constraints were added late in the timeline, but were easily modeled in the IP structure. The not-same-takt task sets in this project were limited to sets of two tasks, and as such, the provided constraint C8 is sufficient. However, if a not-same-takt task set contained more than two tasks, this constraint set can be expanded to take advantage of the clique structure as follows. Let $\bar{\Omega}$ be the set of all not-same-takt sets, each of which is labeled $\Omega_o,\ o \in \{1,\ ..., |\bar{\Omega}|\}$. None of the tasks $i \in \Omega_o$ can be assigned to the same takt, so the following constraint set C8' can be utilized:

$$\sum_{i \in \nu\Omega_o} x_{ikb} \leq 1 \quad \forall\, k \in K,\, b \in W,\ \text{for}\, o \in \{1,\ ..., |\bar{\Omega}|\} \quad \text{C8'}$$

The IP features three distinct zone types: work zones, product zones, and fixed resource coverage zones. The implementation details of these zones, such as their mapping relationships, are easily customizable. It is possible to add, remove, or re-map any of the zoning features with reasonable effort. Such changes would require no alteration of the IP formulation itself, only redefinition of the preprocessing parameters, in which zone relationships are encapsulated.

Implementing task sequencing constraints would require adding new decision variables to the IP to ensure that task start/stop times are properly managed. Adding these variables and associated sequencing constraints to the IP is relatively direct in terms of formulation, but might present significant consequences in terms of runtime. Adding decision variables might always be expected to add runtime, but in particular start/stop time variables are quantified over the real numbers. All other variables currently in the IP are binary, significantly restricting the size of the solution space. Timing variables changes the IP from a BIP to a MILP, and runtime penalties should be expected.

Problem extensions that permit the IP to touch on related production planning problems would necessitate large-scale adaptations to the IP. Examples include extensions to accommodate job sequencing, part logistics, or facility design. The IP does not consider task sequencing, so an infeasibility of this kind may be present in the solution produced by the IP. Through post-processing review, the data sets used did not result in any violations. It is possible to extend the IP to manage task sequencing, and thereby prevent such infeasibilities, but at the cost of introducing a new set of decision variables related to sequencing.

7.1.2. Heuristics

Relative to the IP, the heuristic methods are ill-suited for extensions that add constraints or other gALB features. The first ancestral version of the MRPW heuristic was developed early in the research project, before discovering many of the constraints now represented. Since that time, each constraint added has induced excessive difficulty when adapting the MRPW method

Further, the experiment has shown significant performance problems for the heuristics, in terms of finding feasible solutions and in the quality of those solutions. MRPW-LFI-WZBlock is possibly useful in a few cases. The first is the case of extraordinarily large problem size. The runtime of heuristics scales in a polynomial fashion with respect to problem size and will experience a slower growth rate than the IP. The second scenario for application of heuristic methods is if resources for solving IPs are unavailable.

Acknowledgment

Funding for this research was provided by BMW Manufacturing Co.

Appendix A. Supplementary data

Supplementary material related to this article can be found, in the online version, at doi:https://doi.org/10.1016/j.jmsy.2018.12.011.

References

[1] Scholl A. Balancing and sequencing assembly lines. 2nd ed. Heidelberg: Physica-Verlag.; 1999.
[2] Salveson M. The assembly line balancing problem. J Ind Eng 1955;6(3):18–25.
[3] Wee T, Magazine M. Assembly line balancing as generalized bin packing. Oper Res Lett 1982;1(2):56–9.
[4] Chase R. Survey of paced assembly lines. Ind Eng 1974;6(2):14–8.
[5] Schöniger J, Spingler J. Planung der Montageanlage. Technica 1989;14:27–32.
[6] Milas G. Assembly line balancing… let's remove the mystery. Ind Eng 1990;22:31–6.
[7] Erel E, Sarin S. A survey of the assembly line balancing procedures. Prod Plan Control 1998;9(5):414–34.
[8] Boysen N, Fliedner M, Scholl A. Production planning of mixed-model assembly lines: overview and extensions. Prod Plann Control: Manage Oper 2009;20(5):455–71.
[9] Townsend B. The basics of line balancing and JIT kitting. Boca Raton, FL: Taylor & Francis Group; 2012.
[10] Baybars I. A survey of exact algorithms for the simple assembly line balancing problem. Manage Sci 1986;32:909–32.
[11] Baybars I. An efficient heuristic method for the simple assembly line balancing problem. Int J Prod Res 1986;24:149–66.
[12] Becker C, Scholl A. A survey on problems and methods in generalized assembly line balancing. Eur J Oper Res 2006;183:694–715.
[13] Johnson R. A branch and bound algorithm for assembly line balancing problems with formulation irregularities. Manage Sci 1983;29:1309–24.
[14] Bautista J, Suarex R, Mateo M, Companys R. Local search heuristics for the assembly line balancing problem with incompatibilities between tasks. Proceedings of the 2000 IEEE international conference on robotics and automation 2000:2404–9.
[15] Carnahan B, Norman B, Redfern M. Incorporating physical demand criteria into assembly line balancing. Iie Trans 2001;33:875–87.
[16] Bartholdi J. Balancing two-sided assembly lines: a case study. Int J Prod Res 1993;31:2447–61.
[17] Lee T, Kim Y, Kim Y. Two-sided assembly line balancing to maximize work relatedness and slackness. Comput Ind Eng 2001;40:273–92.
[18] Kim Y, Kim J, Kim Y. Two-sided assembly line balancing: a genetic algorithm approach. Prod Plan Control 2000;11:44–53.
[19] Lapierre SD, Ruiz AD. Balancing assembly lines: an industrial case study. J Oper Res Soc 2004;55:589–97.
[20] Baykasoglu A, Dereli T. Two-sided assembly line balancing using ant-colony-based heuristic. Int J Adv Manuf Technol 2008;36(5-6):582–8.
[21] Simaria A, Vilarinho P. 2-ANTBAL: an ant colony optimization algorithm for balancing two-sided assembly lines. Comput Ind Eng 2009;56:489–506.

[22] Chutima P, Chimklai P. Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimization with negative knowledge. Comput Ind Eng 2012;62:39–55.
[23] Pastor R, Corominas A. Assembly line balancing with incompatibilities and bounded workstations. Ricerca Operativa 2000;30:23–45.
[24] Helgeson W, Birnie D. Assembly line balancing using the ranked positional weight technique. J Ind Eng 1961;12:394–8.
[25] Agrawal P. The related activity concept in assembly line balancing. Int J Prod Res 1985;23:403–21.
[26] Tonge F. Summary of a heuristic line balancing procedure. Manage Sci 1960;7:21–42.
[27] Tonge F. A heuristic program for assembly line balancing. New Jersey: Prentice-Hall; 1961.
[28] Pinto P, Dannenbring D, Khumawala B. A heuristic network procedure for the assembly line balancing problem. Naval Res Logist Rev 1978;25:299–307.
[29] Tonge F. Assembly line balancing using probabilistic combinations of heuristics. Manage Sci 1965;11:727–35.
[30] Arcus A. COMSOAL: a computer method of sequencing operations for assembly lines. Int J Prod Res 1966;4:259–77.
[31] Boysen N, Fliedner M, Scholl A. A classification of assembly line balancing problems. Eur J Oper Res 2007;183:674–93.

Bryan W. Pearce, Ph.D., is a technical process engineering at First Quality. He earned his BS, MS and PhD in Industrial Engineering from Clemson University. His interests are in optimization and risk management for capital project planning and production planning applications."

Kavit Antani, Ph.D. is an assembly quality manager at the BMW Manufacturing plant in Spartanburg, SC responsible for X3, X4, X5, X6 and X7 sports activity vehicles. He earned his BS in Production Engineering from University of Mumbai, India, his MS in Industrial and Systems Engineering from Auburn University and his Ph.D. in Automotive Engineering from Clemson University. His research interests are in automotive assembly, manufacturing complexity and its effects on product quality, and powertrain engineering. He is an active member of SME, SAE and ASQ.

Laine Mears, Ph.D., P.E. is the SmartState Endowed Chair of Automotive Manufacturing at Clemson University. He earned a BS in mechanical engineering from Virginia Tech and MS and Ph.D. degrees in mechanical engineering from Georgia Tech. He is a Fellow of both the American Society of Mechanical Engineers and SME, and associate editor of the ASME Journal of Manufacturing Science and Engineering and the SME Journal of Manufacturing Systems. His research in state estimation and control of manufacturing equipment is sponsored by the National Science Foundation, Office of Naval Research and numerous industrial partners.

Kilian Funk, Ph.D., is a senior software developer with the BMW Group. He earned his diploma in mechanical engineering and his Ph.D. in technical mechanics from the Technische Universität München (TUM). He spent many years with BMW research working on driver assistance systems, mechatronic system design, assembly and production planning methods. His is currently working in the field of autonomous driving.

Maris E. Mayorga, PhD is a Professor in the Fitts Department of Industrial and Systems Engineering at North Carolina State University. She earned her BS in Mechanical Engineering from The George Washington University and her MS and PhD degrees in Industrial Engineering and Operations Research from the University of California, Berkeley. She is a member of IISE and INFORMS. She is an associate editor for *IISE Transactions*, *OMEGA (the International Journal of Management Science)*, *IISE Transactions on Healthcare Systems Engineering, Systems Science and Service Science*, and the *INFORMS Journal on Computing*. Her research interests are modeling and optimization of complex systems under uncertainty, including healthcare and logistics.

Mary E. Kurz, PhD is an associate professor in Industrial Engineering at Clemson University. She earned her BS and MS in Systems Engineering and her PhD in Systems and Industrial Engineering from the University of Arizona. She is a Senior Member of IISE and a member of INFORMS and SME. She is a member of the editorial board for the *Journal of Manufacturing Systems*. Her research interests are in optimization for assembly-related tasks, using exact techniques, heuristics and metaheuristics.