An Ant Colony Optimisation Based Heuristic for Mixed-model Assembly Line Balancing with Setups

Dhananjay Thiruvady
School of Information Technology
Deakin University
Geelong, Australia
dhananjay.thiruvady@deakin.edu.au

Asef Nazari
School of Information Technology
Deakin University
Geelong, Australia
asef.nazari@deakin.edu.au

Atabak Elmi
School of Information Technology
Deakin University
Geelong, Australia
atabak.elmi@deakin.edu.au

***Abstract*—Balancing and sequencing of assembly lines is the process of partitioning the assembly work in terms of operations, and to assign and schedule them to workstations in an optimal way. In particular, in response to highly competitive market conditions, manufacturers face the problem of producing several models of a base product on the same assembly line, which leads to a mixed-model assembly line balancing problem. This problem is proven to be NP-hard and is computationally challenging. In addition to the usual problem constraints (e.g. precedences between operations and satisfying cycle times), we consider setup times between operations, which further complicates the problem. In this work, we present a novel ant colony optimisation approach, which is based on learning permutations of the operations. The permutations are then mapped to an assignment of operations to workstations in a greedy fashion. The numerical experiments demonstrate improvements both in the quality of solutions and significant improvements in computational time in comparison to the exact state of the art solution methods currently available in the literature.**

***Index Terms*—Assembly line balancing; Mixed-model, Setup times, Ant colony optimisation**

I. Introduction

After Henry Ford's great discovery on assembly lines by introducing moving belts for a production system, an assembly line balancing problem (ALBP) dealt with an optimal assignment of operations to workstations (stations or machines) by satisfying several constraints to optimise objective functions including efficiency, cycle time, cost, and so forth to meet customer demand. In this regard, production systems are considered as transformation processes to convert resources into final products including goods and services. With the growing trend and request for greater product variability and shorter life cycles of goods and services, new low demand production systems are replacing the traditional mass production assembly lines to accommodate the recent trends [25]. To find the best configuration for an assembly line design, a huge body of research is devoted to an optimal allocation of operations to workstations [26] and possibly re-optimising an existing configuration [7] in response to some changes.

Assembly lines can be categorised based on the number of product models they process. The ALBPs for a single product with high volumes are referred to as simple ALBP (SALBP) [5]. However, based on competitive market conditions and changing customer demands, a flexible and balanced assembly line is a key indicator for the success of a manufacturing system in rapidly changing business environments. To respond to these conditions, a mixed-model assembly line balancing problem (MMALBP) concerns with customer-centered market requirements on producing a product model with different features or several models on the same production line [24].

An MMALBP is considered as type I (MMALBP-I) when the aim of the problem is to design a new assembly line with known demand. A type II problem concerns with redesigning of an existing line based on some new changes in any component of a given production system [29], [1]. In this paper, a type I version of the problem is under investigation. In other words, the main question in this paper is to find an optimal allocation of operations to workstations with the target of minimizing the number of workstations subject to known production capacity.

Even though MMALBP-I is complicated in its own right, we consider the problem with another dimension of complexity by accounting for setup times. Generally, in the assembly line balancing and sequencing literature, setup times are neglected as they consume a small amount of time in comparison with other operational times. In addition, setup times usually are separated from the main production processes, and they are considered just before or after each operation independently. However, setup times must be brought into account, when for example we deal with a system with a considerably short cycle time [2]. Hence, in this paper, our aim is to solve a mixed-model assembly line balancing problem with setups of type I (MMALBPS-I).

There are several advantages in preferring mixed-model assembly lines over single model assembly lines. In mixed-model assembly lines, one would avoid the construction of several lines and satisfy ever-changing customer demands. In addition, due to producing several models, they have increased flexibility of a manufacturing system. Moreover, they are more suitable and realistic in today's highly competitive global markets. Nevertheless, the assignment of operations to workstations in order to minimize the number of workstations for predefined cycle time and given number of models is NP-hard [10], [5] and finding even a feasible solution (that cannot be guaranteed to be optimal) is time-consuming [4]. Hence, designing an efficient meta-heuristic method for finding an optimal solution for an MMALBPS-I is inevitable while presenting several challenges.

As a brief literature review, extending a SALBP to incorporate setup times was first seen in [3], [22] where the problem was described as a mathematical problem and several meta-heuristics were proposed. An indication of the difficulty of the problem is investigated in [26] where authors formulated the problem in mixed integer programming problem (MIP) format and showed that a standard MIP-solver is not capable of efficiently solving this kind of problems. The MMALBPS is studied as a mathematical model considering product-related inter-operation times in [20], [2] and hybrid meta-heuristics were proposed including ant colonies, bee colonies, genetic algorithms, and so forth [1]. The best treatment of MMALBPS appeared in [2]. They tried to fill the literature gap in providing an efficient exact solution procedure based on Bender's decomposition. Recently, the MMALBPS problem is under further investigation. In particular, considering uncertainty in input data and stochastic sequences of operations are studied in [39], [11]. Also, different variants of this problem including a two-sided assembly line, parallel lines, and robotic two-sided assembly line were considered in [38], [21], [19] by utilising different approaches ranging from mixed integer programming, fuzzy logic, and different hybrid meta-heuristics.

There is a large body of work devoted to using ACO in solving different versions of assembly line balancing and sequencing problems (e.g. [14], [16], [40], [41], [18]). Fattahi et. al. [14] investigate solving a mixed programming model for a multi-manned ALBP using ACO. Kucukkoc and Zhang [16] demonstrate the efficacy of ACO in deadlin with two conflicting objectives in a type-E parallel 2S-ALBP. In addition, different versions of ACO is used in solving SALBP-I [40], SALBP [41], and U-shaped assembly line balancing problem [18].

In order to confront the inherent challenges and difficulties of solving the MMALBPS, we propose an ACO approach. Previous studies with MMALBP have proposed constructive meta-heuristic approaches, which use a neighborhood structure to select an operation for a station [37]. In these studies, a solution is generated by selecting one operation for assignment at a time. This neighborhood structure is typically inefficient when considering other variants of the MMALBP where operations need to be sequenced, as there is the overhead of applying a sequencing procedure after allocating operations. More recent meta-heuristic approaches such as artificial bee colonies [17] and ACO [1], are also build their solutions by assigning jobs to stations. In contrast to the previous approaches for the MMALBPS known in the literature (including ACO), our ACO focuses on learning permutations of operations instead of the operation allocation strategy. After a permutation is constructed, each operation is assigned to a workstation in sequence according to the order in the permutation, in a greedy manner, and preserving feasibility. Numerical experiments show excellent results in finding an optimal solution in short time frames, and often outperforming existing exact approach based on integer programming.

The paper is organised as follows. The MMALBPS problem and its associated mixed integer program is detailed in Section II. Section III discusses the proposed ACO approach and the assignment heuristic for this study. In Section IV we investigate the efficacy of our proposed approach against known integer programming based methods. Section V concludes the paper and provides a discussion about future possibilities related to the variants of the MMALBPS and extension of the ACO approach proposed here.

II. Problem Definition

The aim of a mixed-model assembly line production system is to manufacture different models of a base product with slight changes in some features on a single line. In this problem, a precedence diagram shows the precedence relationship between operations to produce a finished product. Assume there are N operations to complete M models of a product in a manufacturing system with at most S workstations and a given cycle time of length C. The assignment of the operations to workstations (sequencing) follows the precedence diagram by introducing parameters P_{ij}, where $P_{ij} = 1$ if operation i must precede operation j; Otherwise, $P_{ij} = 0$. Also, operation $i \in \{1, \ldots, N\}$ for model $m \in \{1, \ldots, M\}$ has processing time of length T_{im}. An MMALBP with parameters $(N, M, C, P_{ij}, T_{im})$ seeks an optimal assignment of N operations pertaining to M models to workstations so that the number of workstations is minimised. When $M = 1$, we are dealing with SALBP.

Incorporating sequence dependent setup times in an MMALBP problem leads to MMALBPS. As introduced in [27], a setup operation can be categorised as forward or backward operations. A forward setup operation is defined between two consecutive assembling operations of a workstation on the same model. However, a backward setup operation is defined between the last and first assembling operations of a workstation which are performed on two consecutive workpieces. Let's consider μ_{ij}^{m} indicating a forward setup time between operations i and j from the same model, and μ_{ij}^{mk} indicating a backward setup time between operations i and j where operation i belongs to model m and operation j belongs to model k with respect to their precedence diagram. From balancing perspective, or determining the workload of workstations, setup times must be taken into account with processing times and performing orders. In this paper we want to solve an MMALBPS with parameters (N, M, C, P_{ij}, T_{im}, μ_{ij}^{m}, μ_{ij}^{mk}) by designing a new assembly line with a give cycle time C. The final solution determines the minimum number of workstations, the workload of each workstation, and the optimal allocation of operations to the workstations.

A. The Mixed Integer Programming Model of MMALBPS-I

Parameters:

- N: the number of operations and the set $\{1, \ldots N\}$
- M: the number of models and the set $\{1, \ldots M\}$
- S: the maximum number of stations and the set $\{1, \ldots S\}$
- C: cycle time

- T_{im}: process time required for operation i of model m
- $Q_{im} \in \{0,1\}$ is 1 if $T_{im} > 0$
- F_{ijm} forward setup time between operations i and j for model m
- B_{ijmn} backward setup time between operation i of model m and operation j of model n
- $P_{ij} \in \{0,1\}$ is 1 if operation i must precede operation j

Decision Variables:

- $Y_{is} \in \{0,1\}$ is 1 if operation i is assigned to machine s
- $A_s \in \{0,1\}$ is 1 if station s is active
- $w_{ijs} \in \{0,1\}$ is 1 if operation i precedes operation j at machine s
- $X_{ijms} \in \{0,1\}$ is 1 if operation j directly follows operation i of model m at station s
- $Z_{ijmns} \in \{0,1\}$ is 1 if i is the last operation of model m and j is the first operation of model n at station s

Objective function: minimize the number of workstations

$$\text{minimize} \sum_{s=1}^{S} A_s$$

Constraints: subject to

1) assign each operation to exactly one station $\sum_{s=1}^{S} Y_{is} = 1$, for all $i \in N$,
2) guarantee all the predecessors are already assigned to previous machines, for all $i, j \in N, \ i \neq j$
$$\left(\sum_{s=1}^{S} sY_{is} - \sum_{s=1}^{S} sY_{js} \right) P_{ij} \leq 0,$$
3) workstation capacity restriction, for all $s \in S, m, n \in M$
$$\sum_{i=1}^{N} \left(Y_{is}T_{im} + \sum_{j=1}^{N} (X_{ijms}F_{ijm} + Z_{jimns}B_{ijmn}) \right) \leq CA_s,$$
4) the order of active workstations $A_s \geq A_{s+1}$, for all $s \in S$
5) order of the operations and setup operations between them, for all $i, j \in N, m, n \in M,$ and $s \in S$
$$w_{ijs} + w_{jis} + X_{ijms} + X_{jims} + Z_{ijmns} + Z_{jimns} \leq 3(1 - Y_{is} + Y_{js}),$$
$$w_{ijs} + w_{jis} + X_{ijms} + X_{jims} + Z_{ijmns} + Z_{jimns} \leq 3(1 + Y_{is} - Y_{js}), \text{ and}$$
$$w_{ijs} + w_{jis} + X_{ijms} + X_{jims} + Z_{ijmns} + Z_{jimns} \leq 3(Y_{is} + Y_{js}),$$
6) precedence of operations in a station, for all $i, j \in N, i \neq j, s \in S$
$$P_{ij}(Y_{is} + Y_{js}) \leq w_{ijs},$$
7) prevent ordering operations with itself $w_{iis} = 0$ for all $i \in N, s \in S$,
8) ordering within any three operations, for all $i, j, k \in N, i \neq j \neq k, s \in S$
$$w_{iks} + w_{kjs} - 1 \leq w_{ijs},$$
9) operation performing order within a workstation, for all $u \in N, s \in S$
$$\left| \sum_{k=1}^{N} \sum_{l=1}^{N} w_{kls} - \sum_{v|v<u}^{N} v \right| \leq N \left| u - \sum_{p=1}^{N} Y_{ps} \right|,$$
10) each operation in any station has only one successor, for all $i \in N, m \in M$
$$\sum_{j=1}^{N} \sum_{s=1}^{S} X_{ijms} \leq 1,$$
11) only one forward setup operation between any pair of operations, for all $i, j \in N, i \neq j, \ m \in M, \ s \in S$
$$X_{ijms} + X_{jims} \leq 1,$$
12) there is no forward setup operation between any operation and itself, for all $i \in N, \ m \in M$
$$\sum_{s=1}^{S} X_{iims} = 0,$$
13) each workstation has only one backward setup operation, for all $m, n \in M, \ s \in S$
$$\sum_{i=1}^{N} \sum_{j=1}^{N} Z_{ijmns} \leq 1,$$
14) no forward operation if a backward operation is already assigned, for all $i, j \in N, i \neq j, \ m, n \in M, \ s \in S$
$$X_{ijms} \leq 1 - Z_{ijmns},$$
15) determines forward and backward operations in any workstation, for all $i, j \in N, \ m, n \in M, \ s \in S$
$$(Y_{is}Q_{im} + Y_{js}Q_{jn} - 1) - \left(\sum_{k=1}^{N} w_{iks}Q_{km} \right) - \left| \sum_{l=1}^{N} Y_{ls}Q_{ln} - \sum_{p=1}^{N} w_{jps}Q_{pn} - 1 \right| \leq Z_{ijmns},$$
and
$$(Y_{is}Q_{im} + Y_{js}Q_{jm} - 1) - \left| \sum_{k=1}^{N} w_{iks}Q_{km} - \sum_{l=1}^{N} w_{jls}Q_{lm} - 1 \right| \leq X_{ijms}.$$

III. Methods

A. Ant Colony Optimisation

We use Ant colony system (ACS) [13] as the variant of ACO for this study. In a number of studies, ACS has been shown to be very effective, and in particular, this approach has improved characteristics relative to the original ACO method, Ant System [13]. The algorithm is presented in Algorithm 1.

A solution to the problem is represented by a permutation of operations (π). The permutation needs to be mapped to a feasible assignment to workstations, which is represented by $\hat{\pi}$ (an assignment heuristic that allocates operations to stations, including generating precedence feasible schedules, is provided in the next section). The input to the algorithm is an MMALBPS problem instance, a time limit (t_{max}) and a number of solution to be built at each iteration n_s. The first step is to initialise the pheromone trails (Initialise($\mathcal{T}$)), where $\tau_{ij} = \frac{1}{|\mathcal{J}|}$ represents the desirability of picking operation j in position i of π.

Algorithm 1 ACO for MMALBPS

```
1:  Input: An MMALBPS instance, $t_{max}$, $n_s$
2:  Initialise($\mathcal{T}$)
3:  while time elapsed $< t_{max}$ do
4:      for $j = 1$ to $n_s$ do
5:          $\pi^j :=$ ConstructPermutation($\mathcal{T}$)
6:          $\hat{\pi^j} :=$ AssignOperations($\pi^j$)
7:      $\pi^{ib} := \min_{j=1,\ldots,n_{ants}} f(\pi^j)$
8:      $\pi^{bs} :=$ Update($\pi^{ib}$)
9:      $\mathcal{T} :=$ PheromoneUpdate($\pi^{bs}$)
10: return $\pi^{bs}$
```

The main part of the algorithm is between Lines 3–9, where the algorithm executes until a time limit is reached. In each iteration, a number of solutions defined by the input parameter n_s are built by starting with an empty sequence and incrementally adding solution components or operations (ConstructSequence($\mathcal{T}$)). An operation is selected in one of two ways. A random number $q \in (0, 1]$ is generated and if this number is smaller than a predefined parameter q_0, an operation is selected for variable i deterministically. Otherwise the operation is selected probabilistically. Specifically, if $q < q_0$, operation k is chosen for variable i according to:

$$k = \underset{j \in \mathcal{J}}{\operatorname{argmax}} \, \tau_{ij} \cdot \eta_{ij} \tag{1}$$

On the other hand, if $q \geq q_0$, operation k is selected with probability:

$$P(\pi_i = k) = \frac{\tau_{ik} \cdot \eta_{ik}}{\sum_{j \in \mathcal{J}} \tau_{ij} \cdot \eta_{ij}} \tag{2}$$

where η_{ij} is a heuristic bias for selecting operation j given that operation i was the previous selection. For the purposes of the MMALBP problem, we investigated several heuristic biases (e.g. based on setup times μ) and found that $\eta_{ij} = \frac{1}{T_{im}}$ was the most effective.

In the ACS algorithm, a local pheromone update applies to every operation selection (ensures the same selection is less likely to be repeated). In this study, when in position i operation j is chosen, the local pheromone update rule that applies is:

$$\tau_{ij} = \max \tau_{ij} \cdot (1.0 - \rho), \tau_{min} \tag{3}$$

where $\tau_{min} = 0.0001$ is a small value which ensures that no pheromone value becomes too small, thereby always allowing it to be selected.

Once a permutation of the operations has been constructed, a corresponding allocation is produced (AssignOperation(π_j), details provided in the next section). From among the solutions, the iteration best solution is selected (Line 7) by finding the solution using the minimum number of stations. This is followed by updating the global best solution to the iteration best solution if it is an improvement ($\pi^{bs} :=$ Update(π^{ib})) and the final step is to update the pheromone trails ($\mathcal{T} :=$ PheromoneUpdate(π^{bs})). Specifically, the pheromone trails are updated based on the solution components in π^{bs} using:

$$\tau_{ij} = \tau_{ij} \cdot (1.0 - \rho) + \delta \tag{4}$$

where $\delta = Q/f(\pi^{bs})$, and Q is selected such that $0.01 \leq \delta \leq 0.1$. The evaporation rate was set to 0.1 and chosen based on tuning by hand.

B. Assigning Operations to Stations

In previous studies, it has been shown that a permutations can often be mapped to final schedule or assignment efficiently [33], [32], [6], [34]. Hence, for the MMALBPS, we design such a customised heuristic that maps the permutation to a feasible assignment.

The assignment heuristic is presented in Algorithm 2. Given a permutation π of operations, the heuristic assigns the operations to machines satisfying precedences and using up as much of the cycle time as possible at every used station.

Algorithm 2 Placement Heuristic

```
1:  INPUT: $\pi$
2:  $\hat{\pi}^s \leftarrow \emptyset \; \forall s \in S$, $W \leftarrow \emptyset$, $g_i \leftarrow 0 \; \forall i \in S$
3:  for $t \in \pi$ do
4:      $\hat{t} \leftarrow t$
5:      if Prec($t$) not done then
6:          $W \leftarrow W \cup \hat{t}$
7:      else
8:          while $\hat{t} \neq \emptyset$ do
9:              $s :=$ AvailableWorkStation($\hat{t}$)
10:             $(\hat{\pi}, g_s) \leftarrow$ Update($\hat{t}$)
11:             $\hat{t} \leftarrow \emptyset$
12:             for $j \in$ W do
13:                 if Prec($j$) done then
14:                     $\hat{t} \leftarrow j$
15:                     $W \leftarrow W \setminus j$
16:                     break
17: OUTPUT: $\hat{\pi}$
```

As input, the algorithm requires a sequence of the operations. First, $\hat{\pi}$, a waiting list (W) and the station cycle times (g) are initialised. For each operation in π (Line 3), the algorithm attempts to identify a workstation it can be processed by. In Lines 5-6 the operation t is tested to determine if its preceding operations are completed, and if not the operation is placed in the waiting list (W). If the operation is available to be allocated to a station, a feasible station is found (Lines 9-10). This happens by first determining which (first) machine an operation can be allocated to (Line 9) and then update the assignment in $\hat{\pi}$ and ensure the relevant times are added to g_s (Line 10). That is, the time includes the processing time of an operation (p_t), the forward setup time from the last operation $\bar{t}$ to the current operation $\hat{t}$ $(\mu^{mk}_{\bar{t}\hat{t}})$ and the backward setup time from the current operation $\hat{t}$ to the first operation $(\hat{t}^s_0)$ on the machine $\mu^{mk}_{\hat{t}\hat{t}^s_0}$. Once the operation is allocated, the waiting list is examined to determine if any of these operations are precedence-freed (i.e. preceding operations are complete), and if so, this operation is allocated (Lines 12–16) as previously described. At the end of this procedure, all operations have been allocated, satisfying precedences, ensuring the cycle times are feasible, and the final allocation $\hat{\pi}$ is output by the algorithm.

IV. Experiments and Results

ACS for the MMALBPS was implemented in C++ compiled in GCC-5.4.0. The experiments were all conducted on Monash University's Campus Cluster, MonARCH.[1]

The comparison in this study is to that of [2], which is the current state-of-the-art for the MMALBPS. The problem instances were also obtained from this study, and we compare solutions for single model and two model problems. There is no set of standard benchmark problems containing sequence dependent setup times. The set of problems solved in [2] had been generated using the existing precedence diagrams in the literature where, the assembling and setup operation times had been produced randomly. Additionally, to have an efficient comparison, the proposed Bender's decomposition (BDA) approach proposed by [2] been implemented by and utilized for solving the same set of problems. The problem instances, number of assembling operations and cycle times are provided in Table I. There are 20 problem instances per model type and we run ACS 30 times on each problem instance.

The parameters settings for ACS were found through tuning by hand on a subset of the problem instances. The values $\{0.2, 0.1, 0.0.5, 0.01\}$ were tested for ρ and it was found that 0.1, was the most effective value. Values of $\{0.5, 0.2, 0.1, 0.01\}$ were tested for q_0 and it was found that $q_0 = 0.1$ was best. As previously mentioned, $\tau_{min} = 0.0001$ was used to ensure the probability of selecting a task in a position is never 0.

The first comparison is for the single model problems. These results are presented in Table II. We see that for all

[1]https://confluence.apps.monash.edu/display/monarch/MonARCH+Home

TABLE I: Main characteristics and numbers of the test instances.

Problem Name	**N**	**C**	**Instance Number**	
			M=1	**M=2**
Bowman	8	10	1	21
Jackson	11	10	2	22
[23]	12	10	3	23
[28]	14	10	4	24
[9]	15	10	5	25
[15]	16	10	6	26
[30]	17	10	7	27
[35]	19	10	8	28
Mitchell	21	10	9	29
[36]	25	10	10	30
Heskiaoff	28	10	11	31
Buxey	29	10	12	32
Sawyer	30	10	13	33
Lutz1	32	10	14	34
Gunther	35	10	15	35
Kilbridge	45	10	16	36
Hahn	53	10	17	37
Warnecke	58	10	18	38
Tonge	70	10	19	39
Arcus 1	83	10	20	40

small problem instances ACS performs as well as the integer programming model (IP) or BDA. For the larger problem instances (Instance 10 or greater), the IP model is intractable, always running out of memory. For these cases, ACS is the best performing method nearly always equal to BDA or better. The only problem instance where ACS struggles is Instance 19, where BDA finds an excellent solution, but requires a full hour of run-time to do so.

On examining the results further, we see that the standard deviation of ACS across several runs is low (0 for most problem instances), demonstrating the reliability of the algorithm in finding good solutions. Furthermore, the run times needed by ACS are substantially lower than that of BDA or IP.

Table III shows the results for the two model problem instances. The pattern observed here is very similar to that of the single model case, though, ACS is consistently more effective in this case compared to BDA, even for the large problem instances. The run time requirements of ACS are low and the standard deviations show that very high-quality solutions can be found reliably across several runs of the algorithm.

Figures 1 and 2 provide a visualisation of the results seen in the tables. Figure 1 shows the percentage difference of each method relative to the best solution found between all of these methods. For example, the values for each instance for ACS is computed as $\frac{|ACS-UB^*|}{UB^*}$, where $UB^* = \min(IP, BDA, IP)$. We see that ACO nearly always performs best, except on the largest problem instances for single or two models.

Figure 2 shows how the run times of ACS and BDA, for single model (left) and two model (right), vary in finding their best known solutions. Note that the scales in these figures, for each method, are significantly different. BDA clearly requires increased times with increasing problem size. This is partially true for ACS, which generally requires more time for more complex problems, but is generally quite small.

TABLE II: A comparison of integer programming (IP), Bender's decomposition (BDA) and ACS for single model problems; The best values found by each algorithm is reported as Opt. Val. and the best values are marked in bold; ACS is run 30 times per instance, hence, the objective value's associated standard deviations (SD) are also reported; For ACS, the cycle time used is reported as Cyc. Time.

	IP		BDA		ACS			
Instance	Best	Time	Best	Time	Best	SD	Time	Cyc. Time
1	7	0.21	7	0.09	7	0	0	935
2	7	4.84	7	0.48	7	0	0	970
3	8	2.76	8	0.19	**6**	0	0.03	985
4	10	10.18	10	0.42	**9**	0	0	975
5	7	156.26	7	0.28	7	0	0	987
6	9	27.40	9	0.24	9	0	0	979
7	13	300.54	13	0.77	13	0	0	975
8	13	412.57	13	0.83	**12**	0	0.02	994
9	14	41.31	14	0.44	14	0	0	986
10	out mem	-	13	1.74	13	0	0	987
11	out mem	-	17	3.69	17	0	0	997
12	out mem	-	16	5.59	**14**	0	6.09	998
13	out mem	-	20	19.93	**19**	0	6.08	1000
14	out mem	-	19	3.27	**18**	0	0.01	998
15	out mem	-	25	6.26	25	0	0	991
16	out mem	-	29	54.38	29	0	0.22	997
17	out mem	-	31	36.62	**30**	0	4.09	995
18	out mem	-	34	178.78	34	0	0.01	999
19	out mem	-	**36**	3600	39	0	0.18	1000
20	out mem	-	-	-	**53**	0	0.87	999

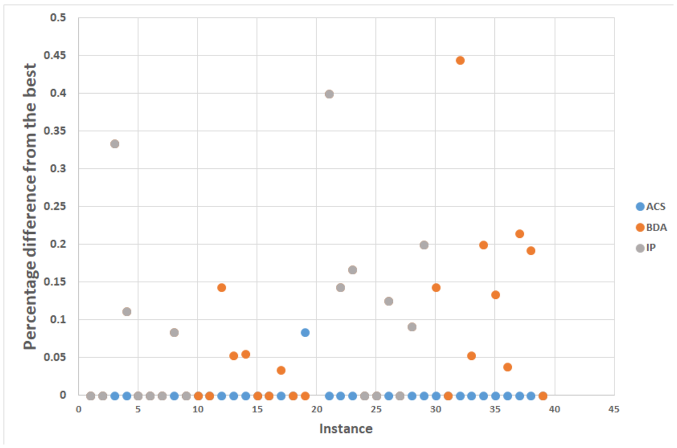

Fig. 1: The solution quality of each method relative to the best solution found by IP, BDA and ACS.

V. Conclusion and Future Work

In this paper, we investigate an ant colony optimisation approach for tackling the mixed model assembly line balancing problem with setups. In contrast to previous ACO approaches for this problem, we propose a model where the ACO learns a permutation of the operations. After a permutation is constructed, each operation is assigned to a station in sequence according to the order in the permutation, greedily, an ensuring feasibility. We find that this approach leads to excellent results in short time frames, often outperforming existing exact approach based on integer programming.

A key aspect of future work, would be to determine how the proposed approach in this study can be adapted to other variants of the problem [38], [21], [19], including also the stochastic variants [39], [11]. For example, this ACS method can be adapted in a straightforward manner to MMALBPS where the objective is to minimise cycle time rather than number of workstations. A further extension would be to consider a multi-objective variant which aims to minimise the number of workstations and also cycle time.

While we have shown that the proposed ACS approach is very effective, there is still room for improvement. Heuristics or meta-heuristics on their own are unable to provide guarantees on the quality of solutions. A potential solution to this is to develop matheuristics which hybridise ACO with integer programming (e.g. the study by [31]). Such approaches can provide guarantees and also provide significant improvements in solution quality, especially for large problems.

The problem instances considered here are relatively small and we are working to develop a problem instance set that reflects real world situations more closely. We are currently in the process of designing tests cases in conjunction with experts who are working assembly lines problems in the industry. In such cases, scalibility of the algorithms (matheuristics) can be improved with parallel implementations, similar to the studies [8], [12].

References

[1] Şener Akpinar and Adil Baykasoğlu. Modeling and solving mixed-model assembly line balancing problem with setups. part i: A mixed integer linear programming model. *Journal of manufacturing systems*, 33(1):177–187, 2014.

[2] Sener Akpinar, Atabak Elmi, and Tolga Bektaş. Combinatorial benders cuts for assembly line balancing problems with setups. *European Journal of Operational Research*, 259(2):527–537, 2017.

TABLE III: A comparison of integer programming (IP), Bender's decomposition (BDA) and ACS for two model problems; The best values found by each algorithm is reported as Opt. Val. and the best values are marked in bold; ACS is run 30 times per instance, hence, the objective value's associated standard deviations (SD) are also reported; For ACS, the cycle time used is reported as Cyc. Time.

	IP		BDA		ACS			
Instance	Best	Time	Best	Time	Best	SD	Time	Cyc. Time
21	7	0.68	7	0.63	**5**	0	0	972
22	8	5.84	8	0.14	**7**	0	0	984
23	7	3.26	7	0.11	**6**	0.48	5.46	977
24	8	103.57	8	0.13	8	0	0	983
25	7	182.17	7	0.26	7	0	0	963
26	9	72.74	9	0.41	**8**	0	0	970
27	9	988.36	9	0.36	9	0	0	987
28	12	163.08	12	0.67	**11**	0	0	982
29	12	459.71	12	0.25	**10**	0	0.04	985
30	out mem	-	16	0.72	**14**	0	0	981
31	out mem	-	14	5.37	14	0	0	991
32	out mem	-	13	3.15	**9**	0	0.01	996
33	out mem	-	20	3.26	**19**	0	0.01	995
34	out mem	-	18	3.03	**15**	0	0.01	990
35	out mem	-	17	8.80	**15**	0	0.01	994
36	out mem	-	27	42.57	**26**	0.44	11.34	990
37	out mem	-	34	103.05	**28**	0	2.83	999
38	out mem	-	31	209.85	**26**	0.3	6.78	993
39	out mem	-	39	3600	39	0.48	16.27	998
40	out mem	-	n	n	**38**	0.3	10.83	996

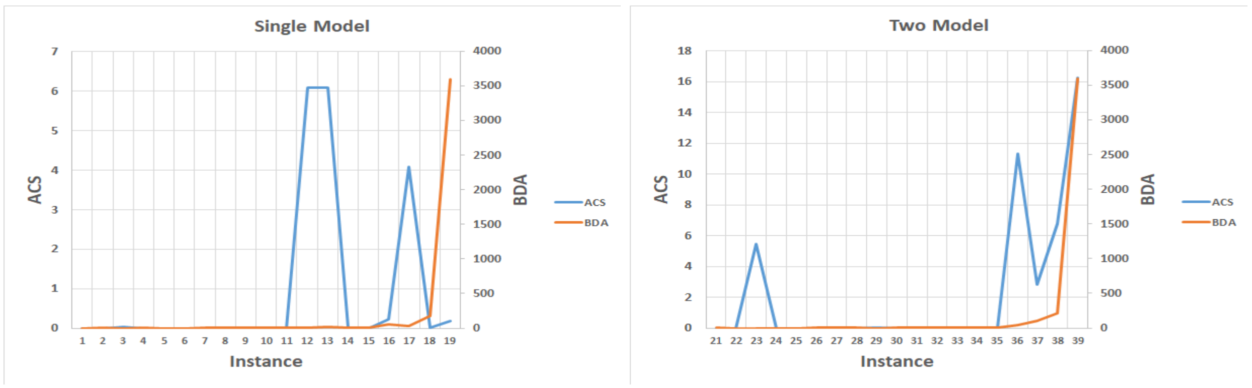

Fig. 2: The run times to find the best known solutions for BDA and ACS on single model MMALBPS (left) and two model MMALBPS (right).

[3] Carlos Andres, Cristobal Miralles, and Rafael Pastor. Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research*, 187(3):1212–1223, 2008.

[4] Olga Battaïa and Alexandre Dolgui. Reduction approaches for a generalized line balancing problem. *Computers & Operations Research*, 39(10):2337–2345, 2012.

[5] Olga Battaïa and Alexandre Dolgui. A taxonomy of line balancing problems and their solutionapproaches. *International Journal of Production Economics*, 142(2):259–277, 2013.

[6] C. Blum, D. Thiruvady, A. T. Ernst, M. Horn, and G. Raidl. A Biased Random Key Genetic Algorithm with Rollout Evaluations for the Resource Constraint Job Scheduling Problem. In J. Liu and J. Bailey, editors, *AI 2019: Advances in Artificial Intelligence*, pages 549–560, Cham, 2019. Springer International Publishing.

[7] Nils Boysen, Malte Fliedner, and Armin Scholl. Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research*, 192(2):349–373, 2009.

[8] O. Brent, D. Thiruvady, A. Gmez-Iglesias, and R. Garcia-Flores. A Parallel Lagrangian-ACO Heuristic for Project Scheduling. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2985–2991, 2014.

[9] Joseph Bukchin, Ezey M Dar-El, and Jacob Rubinovitz. Mixed model assembly line design in a make-to-order environment. *Computers & Industrial Engineering*, 41(4):405–421, 2002.

[10] Yossi Bukchin and Ithai Rabinowitch. A branch-and-bound based solution approach for the mixed-model assembly line-balancing problem for minimizing stations and task duplication costs. *European Journal of Operational Research*, 174:492–508, 2006.

[11] Zeynel Abidin Çıl, Süleyman Mete, Eren Özceylan, Zülal Diri Kenger, and Hande Çil. A heuristic algorithm for assembly line balancing with stochastic sequence-dependent setup times. In *2017 8th International Conference on Information Technology (ICIT)*, pages 424–428. IEEE, 2017.

[12] D. Cohen, A. Gómez-Iglesias, D. Thiruvady, and A. T. Ernst. Resource Constrained Job Scheduling with Parallel Constraint-Based ACO. In Markus Wagner, Xiaodong Li, and Tim Hendtlass, editors, *Proceedings of ACALCI 2017 – Artificial Life and Computational Intelligence*, volume 10142 of *Lecture Notes in Computer Science*, pages 266–278.

Springer International Publishing, 2017.

[13] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, Massachusetts, USA, 2004.

[14] Parviz Fattahi, Abdolreza Roshani, and Abdolhassan Roshani. A mathematical model and ant colony algorithm for multi-manned assembly line balancing problem. *The International Journal of Advanced Manufacturing Technology*, 53(1-4):363–378, 2011.

[15] José Fernando Gonçalves and Jorge Raimundo De Almeida. A hybrid genetic algorithm for assembly line balancing. *Journal of Heuristics*, 8(6):629–642, 2002.

[16] Ibrahim Kucukkoc and David Z Zhang. Type-e parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimisation based approach with optimised parameters. *Computers & Industrial Engineering*, 84:56–69, 2015.

[17] Ibrahim Kucukkoc and David Z Zhang. Mixed-model parallel two-sided assembly line balancing problem: A flexible agent-based ant colony optimization approach. *Computers & Industrial Engineering*, 97:58–72, 2016.

[18] Ming Li, Qiuhua Tang, Qiaoxian Zheng, Xuhui Xia, and CA Floudas. Rules-based heuristic approach for the u-shaped assembly line balancing problem. *Applied Mathematical Modelling*, 48:423–439, 2017.

[19] Zixiang Li, Mukund Nilakantan Janardhanan, Qiuhua Tang, and SG Ponnambalam. Model and metaheuristics for robotic two-sided assembly line balancing problems with setup times. *Swarm and Evolutionary Computation*, 50:100567, 2019.

[20] Ehsan Nazarian, Jeonghan Ko, and Hui Wang. Design of multi-product manufacturing lines with the consideration of product change dependent inter-task times, reduced changeover and machine flexibility. *Journal of Manufacturing Systems*, 29(1):35–46, 2010.

[21] Uğur Özcan. Balancing and scheduling tasks in parallel assembly lines with sequence-dependent setup times. *International Journal of Production Economics*, 213:81–96, 2019.

[22] Rafael Pastor, Carlos Andrés, Cristóbal Miralles, et al. Corrigendum to balancing and scheduling tasks in assembly lines with sequence-dependent setup[european journal of operational research 187 (2008) 1212–1223]. *European Journal of Operational Research*, 201(1):336, 2010.

[23] SG Ponnambalam, P Aravindan, and G Mogileeswar Naidu. A comparative evaluation of assembly line balancing heuristics. *The International Journal of Advanced Manufacturing Technology*, 15(8):577–586, 1999.

[24] Reza Ramezanian and Abdullah Ezzatpanah. Modeling and solving multi-objective mixed-model assembly line balancing and worker assignment problem. *Computers & Industrial Engineering*, 87:74–80, 2015.

[25] Armin Scholl. *Balancing and Sequencing of Assembly Lines*. Contributions to Management Science. Physica-Verlag HD, 1999.

[26] Armin Scholl, Nils Boysen, and Malte Fliedner. The sequence-dependent assembly line balancing problem. *OR Spectrum*, 30(3):579–609, 2008.

[27] Armin Scholl, Nils Boysen, and Malte Fliedner. The assembly line balancing and scheduling problem with sequence-dependent setup times: problem extension, model formulation and efficient heuristics. *OR spectrum*, 35(1):291–320, 2013.

[28] Ana S Simaria and Pedro M Vilarinho. 2-antbal: An ant colony optimisation algorithm for balancing two-sided assembly lines. *Computers & Industrial Engineering*, 56(2):489–506, 2009.

[29] Ana Sofia Simaria and Pedro M Vilarinho. A genetic algorithm based approach to the mixed-model assembly line balancing problem of type ii. *Computers & Industrial Engineering*, 47(4):391–407, 2004.

[30] Ping Su and Ye Lu. Combining genetic algorithm and simulation for the mixed-model assembly line balancing problem. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 4, pages 314–318. IEEE, 2007.

[31] D. Thiruvady, G. Singh, and A. T. Ernst. *Hybrids of Integer Programming and ACO for Resource Constrained Job Scheduling*, pages 130–144. Springer International Publishing, Cham, 2014.

[32] D. Thiruvady, M. Wallace, H. Gu, and A. Schutt. A Lagrangian Relaxation and ACO Hybrid for Resource Constrained Project Scheduling with Discounted Cash Flows. *Journal of Heuristics*, 20(6):643–676, 2014.

[33] D. R. Thiruvady, B. Meyer, and A. T. Ernst. Strip Packing with Hybrid ACO: Placement Order is Learnable. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1207–1213, 2008.

[34] Dhananjay Thiruvady, Irene Moser, Aldeida Aleti, and Asef Nazari. Constraint programming and ant colony system for the component deployment problem. *Procedia Computer Science*, 29:1937–1947, 2014.

[35] Nick T Thomopoulos. Mixed model line balancing with smoothed station assignments. *Management science*, 16(9):593–603, 1970.

[36] Pedro M Vilarinho and Ana Sofia Simaria. A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations. *International Journal of Production Research*, 40(6):1405–1420, 2002.

[37] Pedro M Vilarinho and Ana Sofia Simaria. Antbal: an ant colony optimization algorithm for balancing mixed-model assembly lines with parallel workstations. *International journal of production research*, 44(2):291–303, 2006.

[38] Wucheng Yang and Wenming Cheng. Modelling and solving mixed-model two-sided assembly line balancing problem with sequence-dependent setup time. *International Journal of Production Research*, pages 1–22, 2019.

[39] Behzad Zahiri, Reza Tavakkoli-Moghaddam, and Mohammad Rezaei-Malek. An mcda-dea approach for mixed-model assembly line balancing problem under uncertainty. *Journal of Intelligent & Fuzzy Systems*, 30(5):2737–2748, 2016.

[40] Qiao Xian Zheng, Yuan Xiang Li, Ming Li, and Qiu Hua Tang. An improved ant colony optimization for large-scale simple assembly line balancing problem of type-1. In *Applied Mechanics and Materials*, volume 159, pages 51–55. Trans Tech Publ, 2012.

[41] Yu-guang Zhong and Bo Ai. A modified ant colony optimization algorithm for multi-objective assembly line balancing. *Soft Computing*, 21(22):6881–6894, 2017.