An Adaptive Ant Colony Optimization for Solving Assembly Line Balancing Problem

Fuping Deng[1], Chaoyong Zhang[1], Kunlei Lian[1], and Shaotan Xu[1]

[1]State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan, 430074

Abstract

An adaptive ant colony optimization algorithm is proposed in this paper to solve the assembly line balancing problem (ALBP). Considering the characteristics of the ALBP, a feasible solution construction strategy as well as an improved objective function that performs superior to traditional methods in evaluating solution quality are developed. In addition, the improved algorithm utilizes a strategy to adaptively adjust algorithmic parameters, including pheromone evaporation rate, to address the issue of getting stuck in local optima and slow convergence rate that are often observed in traditional ant colony optimization algorithms. Finally, performance of the proposed algorithms is validated against state-of-the-art algorithms on benchmarking instances, and experimental results show the feasibility and effectiveness of the proposed algorithm.

Key words: Assembly line balancing; ant colony optimization (ACO); adaptive; artificial intelligence

1 Introduction

Assembly line is a widely used component in manufacturing environments, and various challenges emerge in its design and daily operations, among which the assembly line balancing problem (ALBP) exists as one of most significant ones. It is noted that a small improvement towards ALBP can lead to significant efficiency enhancement and cost reduction (Salveso, 1955); On the other hand, ALBP is a classical NP-hard combinatorial optimization problem in which the complexity increases exponentially with the number of jobs and there yet exist polynomial-time algorithms to obtain optimal solutions in reasonable computation times (Kilincci, 2010). Three solution strategies towards solving the ALBP can be found in the literature, namely, exact algorithms, heuristic algorithms and artificial intelligence algorithms. Exact algorithms are able to find

optimal solutions, but often require tremendous computation power and time. Due to these limitations, they are mainly used to solve small-sized problems and can hardly be applied in real-world production systems (Peeters and Degraeve, 2006; Scholl and Becker, 2006). Heuristic algorithms have received many attentions from the research community due to its implementation simplicity; however, they generally take long time to identify optimal solutions and also it is often hard to verify whether the solutions they find are optimal or not (Ponnambalam et al., 1999). Artificial intelligence algorithms, including genetic algorithm, simulated annealing and tabu search, witness significant advancements in recent years and have been applied to solve the ALBP successfully (O. et al., 2011; Ozcan and Toklu, 2008).

Ant colony optimization (ACO) is an intelligent optimization algorithm proposed by Clolrni (Colorni et al., 1991) in 1991 and has been applied to various combinatorial optimization problems. Bautista and Pereira (2002) made the first attempt to solve a simple assembly line balancing problem using ant colony algorithms based on an ant system, but the optimization results were not optimal. McMullen and Tarasewich (2003) obtained superior ACO performance on a number of benchmarking instances in solving a more complex assembly balancing problem that is characterized by mixed job types, stochastic processing times and parallel workstations. Bautista and Pereira (2007) studied an assembly balancing problem with timing and spatial constraints and explored its solution method based on ant colony algorithms.

The aforementioned researches in solving assembly balancing problems using ACO suffer various performance issues when compared with other algorithms in the literature. Some of them employ an oversimplified pheromone updating strategy that jeopardizes ACO's ability to converge to optimal solutions. Others use simple objective functions that often fail to correctly evaluate solution qualities, which results in inferior performance in identifying promising solutions. To address these issues, we propose an adaptive ACO to solve the assembly line balancing problem. It tries to avoid local optima by utilizing both external and historical information to adaptively adjust global pheromone evaporation factor in the process of ant path construction. In addition, the algorithm incorporates balancing and smoothing factors in the objective function, which improves ACO's ability in identifying promising solutions. Performance of the proposed algorithm is validated on benchmarking instances.

2 Assembly Line Balancing and Mathematical Model

ALBP refers to the assignment of finite set of jobs to finite set of workstations in order to maximize workstation utilization, minimize overall overload time and minimize balancing objective value, subject to given processing constraints as well as cycling requirements. It involves the coordination of various processes within an assembly line and needs to address the inconsistency in process machining times. Assembly line

productivity as well as product quality are both greatly affected by its balancing level.

The ALBP this paper aims to address can be mathematically described as follows:

$$\max. \quad f(m) = \lambda L - I \tag{1}$$

$$\text{s.t.} \quad L = \frac{\sum_{i=1}^{n} t_i}{mC} \tag{2}$$

$$I = \sqrt{\frac{\sum_{k=1}^{m}(max(T(S_k)) - T(S_k))^2}{m}} \tag{3}$$

$$S_i \cap S_j = \emptyset,\ i, j = \{1, 2, \cdots\}, i \neq j \tag{4}$$

$$\cup_k S_k = E,\ \forall i \in S_x, j \in S_y, 1 \leq x, y \leq n, x \leq y \text{ if } P_{ij} = 1 \tag{5}$$

$$T(S_k) \leq C \tag{6}$$

In this model, L is the balancing rate of the assembly line; I is the smoothing factor; λ is a user-defined parameter and $\lambda > 1$; E is the set of jobs within the assembly line; S_k is the set of jobs assigned to workstation k; C is the assembly line cycle; t_i is the processing time of job i; $T(S_k)$ is the total processing time of workstation k; P is the precedence matrix of ALBP and $P = [P_{ij}]_{n \times n}$, $P_{ij} = 1$ job i must be processed right before job j, 0 otherwise.

3 The Adaptive ACO for the ALBP

3.1 Solution construction strategy

In order to use ACO to solve the ALBP, the jobs and connections between jobs and workstations can be viewed as the nodes and arcs on a graph that are to be traversed by ACO ants. The assignment process of jobs to workstations can then be seen as the process of ants traveling through the graph with the guidance of pheromone and heuristic information.

The solution construction is a key step in employing ACO to solve ALBP and this paper creates a feasible solution by sequentially assigning available jobs to corresponding workstations. To this end, we define the following notations:

- Unassigned task: a task that hasn't been assigned to any workstation yet
- Available task: a task that hasn't yet been assigned but satisfies precedence constraints between tasks, also all of its proceeding tasks have been assigned
- Assignable task: an available task that satisfies cycling constraints.

The feasible solution construction algorithm can be described as follows:

1. Open a workstation
2. Identify the set of available tasks from all unassigned tasks and the associated precedence constraints among tasks, if the unassigned task set is empty, go to step 7
3. Identify the set of assignable tasks from available tasks and cycling constraints
4. If the set of assignable tasks is empty, go to step 6
5. Select a task from the set of assignable tasks according to defined rules and assign it to the current workstation, go to step 2
6. Open a new workstation, go to step 3
7. Stop

Using the way defined in the above algorithm to identify the set of assignable tasks, there always exists an optimal assignable task in the set.

One key characteristic of ACO is its utilization of pheromone feedback and heuristic information during its search for global optimality; therefore, the way of pheromone updating and heuristic information selection have a huge impact on the performance of ACO. In this paper, we define τ_{ij} as the pheromone intensity on arc (i, j) traversed by ants and represents the expectation of assigning task i to workstation j. The corresponding heuristic information is computed by static precedence rules. In addition, the heuristic information of ACO consists of the maximal task completion time and maximal number of succeeding tasks, and the visibility η_i of task i can be computed as

$$\eta_i = \frac{t_i}{C} + \frac{U_i}{max_{i=1,2,\cdots,N} U_i} \tag{7}$$

where U_i is the number of succeeding tasks of task i.

In order to improve ACO's search capability and avoid stagnating into local optima, we propose a hybrid search strategy by which an ant chooses task i and assigns it to workstation j:

$$i = \begin{cases} I_1 & \arg max_{i \in N_j} (\tau_{ij}(t))^{\alpha} (\eta_i)^{\beta}, 0 \le r \le r_1 \\ I_2 & p_{ij} = \frac{(\tau_{ij}(t))^{\alpha} (\eta_i)^{\beta}}{\sum_{z \in N_j} (\tau_{zj}(t))^{\alpha} (\eta_z)^{\beta}}, r_1 < r \le r_1 + r_2 \\ I_3 & \text{random select } i \in N, r_1 + r_2 < r \le r_1 + r_2 + r_3 \end{cases} \tag{8}$$

where r is a random number chosen from (0, 1); r_1, r_2, r_3 are user-defined parameters and $0 \le r_1, r_2, r_3 \le 1, r_1 + r_2 + r_3 = 1$; N_j is the set of assignable tasks for ant i on the current workstation j; z is an assignable

task; and α, β are two key parameters deciding pheromone intensity and heuristic information.

This hybrid search strategy probabilistically chooses one of the three strategies: 1) utilization: choose with probability r_1 from assignable tasks N_j the task with maximal $(\tau_{ij}(t))^{\alpha}(\eta_i)^{\beta}$ to assign to workstation j; 2) exploration: choose with probability r_2 that is given by I_2 in the equation a task; 3) random selection: choose with probability r_3 a task from the set of assignable tasks.

Following the aforementioned strategy, save the assignable task chosen by ant k into table $tabu_k$ and when all n tasks have been added to $tabu_k$, ant k has finished on traversal of the search graph and the resulting task sequence is a feasible solution to the underlying problem.

3.2 Objective function

The quality of a constructed solution following the strategy described in the previous section must be evaluated and this section defines the objective function for this purpose. The type 1 assembly line balancing problem (ALBP-1) aims to minimize the number of workstations given a fixed cycle time; However, many solutions tend to have the same objective value if we use the number of workstations m as objective function and it is therefore difficult to identify good solutions. In order to facilitate the pheromone accumulation during ants' searching process, this paper uses the following objective function:

$$\max f(m) = \lambda L - I = \lambda \frac{\sum_{i=1}^{n} t_i}{mC} - \sqrt{\frac{\sum_{k=1}^{m}(max(T(S_k)) - T(S_k)^2}{m}} \tag{9}$$

This function employs both assembly line balancing rate L and smoothing factor I to evaluate the quality of an assembly line balancing solution. Using this objective function, an assembly line with better balance has higher L value and smaller I value. We give L a bigger weight $\lambda > 1$ considering the comparative importance of L and I. This objective function improves ACO's capability to identify better solutions from solutions with the same number of workstations, enhances the learning and updating of pheromones and speeds up the algorithm's convergence rate.

3.3 Pheromone updating strategy

The pheromone updating strategy in this paper utilizes a combination of local pheromone updating and global pheromone updating from ant colony system.

- Local pheromone trail updating: when the algorithm constructs a feasible solution, the pheromone on arc (i, j) is updated using the following equation after an ant assigns task i to workstation j:

$$\tau_{ij}(n) = (1 - \rho_1)\tau_{ij}(n-1) + \rho_1\tau_0 \tag{10}$$

where ρ_1 is the evaporation factor of local pheromone and $0 \leq \rho_1 \leq 1$; τ_0 is the initial pheromone level.

Local pheromone trail updating aims to reduce the impact of assigned tasks on following ants traversing the same arc and improve the search capability of un-explored arcs, which increases the search space of the algorithm in order to explore those areas that contain the optimal solution.

- Global pheromone trail updating: after all the ants finish traversal, only the solution with the best objective value will be used to update the pheromone, which increases the search capability of ACO. In the current iteration, the best ant updates the global pheromone using the following equation:

$$\tau_{ij}(t) = (1 - \rho_2)\tau_{ij}(t-1) + \rho_2 \triangle \tau_{ij}^{gb}(t) \tag{11}$$

where

$$\triangle \tau_{ij}^{gb}(t) = \begin{cases} f(m) & (i,j) \text{ belongs to the currrent best solution} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

and ρ_2 is the global pheromone evaporation factor, $0 \leq \rho_2 \leq 1$ and $f(m)$ is the objective value of the current optimal solution.

Using the aforementioned global pheromone trail updating strategy, the pheromone volume that belongs to the current best solution will be higher than other solutions after a few search iterations. This can increase the convergence speed but might also suffer from local optima. In order to improve ACO's global searching capability, this paper proposes a strategy to adaptively modify the value of ρ_2 in order to prevent the pheromone intensity of certain arcs from getting too low or too high. We set the initial value of ρ_2 as $\rho_2(t_0) = 1$ and decrease its value when the best objective value doesn't improve after N iterations:

$$\rho_2(t) = \begin{cases} 0.95\rho_2(t-1) & 0.95\rho_2(t-1) \geq \rho_{2min} \\ \rho_{2min} & \text{otherwise} \end{cases} \tag{13}$$

where ρ_{2min} is the minimal value of ρ_2 and is used to prevent ρ_2 from getting too small during the search process, which may lead to slow convergence speed of the overall algorithm.

3.4 Algorithm workflow

The overall algorithm works as follows

1. Initialize algorithm parameters

2. Open a new workstation and create the set of initial available tasks and assignable tasks

3. conduct the following steps for every ant

   (a) Initialize the pheromone value for every arc

   (b) Choose a task from the set of assignable tasks using formula (8) and assign it to the current open workstation

   (c) Add the assigned task into table *tabu*

   (d) Update the pheromone according to formula (10)

   (e) Move to the next task and open a new workstation if the current open workstation is full

   (f) Create new set of available tasks and assignable tasks

   (g) Repeat the above steps until all tasks are assigned

4. Compute the objective value for every ant using formula (9) and identify the best solution, update the global best solution if the current best solution is better

5. Update the global pheromone trail evaporation factor using formula (13)

6. Update the global pheromone trail using formula (11)

7. Set $N_c = N_c + 1$;

   (a) If $N_c > N_{C_{max}}$, then output the best solution and stop

   (b) Clear the table *tabu* for all ants and go to step 2.

4 Computational Experiments

Based on the descriptions in previous sections, we validate the performance of the proposed ACO on various benchmark instances. The algorithm is implemented in C++ on a Windows XP machine with Pentium IV 2.20GHz and 2G memory. Inasmuch as parameter settings have big impacts of ACO's efficiency, we identify the parameters using multiple trials and the final parameters are set as follows: the number of ants (N_ANT_COUNT) = number of tasks (N_TASK_COUNT), the maximum number of iterations $N_{C_{max}} = 50$, $\tau_0 = 1$, $\alpha = 1.0$, $\beta = 2.0$, $\rho_1 = 0.1$, $\rho_2(t_0) = 1$, $\rho_{2_{min}} = 0.5$, $r_1 = 0.6$, $r_2 = 0.3$, $r_3 = 0.1$.

We first analyze the proposed ACO's performance based on the results for problem $C = 57$ from Kilbridge benchmark set, then present the overall performance comparison for all 24 benchmark problems in the set. The task precedence graph and computational results for problem Kilbridge are given in figure 1 and table 1, respectively.

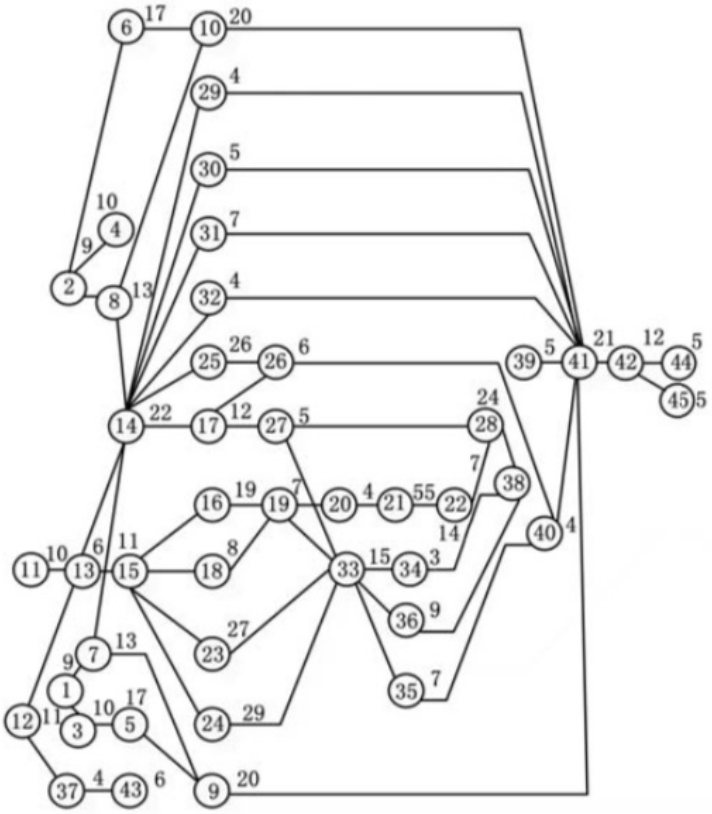

Figure 1: Task precedence graph of problem Kilbridge

Table 1: Computational result of problem Kilbridge

Workstation No.	**Tasks**	**Workstation Time(s)**	**Free Time (s)**
1	1, 11, 12, 13, 15, 18, 39	55	2
2	2, 7, 8, 16	54	3
3	14, 17, 19, 20, 27, 31	57	0
4	21	55	2
5	23, 24	56	1
6	3, 4, 22, 30, 33, 34	57	0
7	5, 25, 29, 36	56	1
8	6, 26, 28, 35	54	3
9	9, 10, 32, 38, 40	55	2
10	37, 41, 42, 43, 44, 45	53	4

It can be seen from the computational results that our proposed algorithm is able to identify the solution with minimal number of workstations and balance rate of 96.8% for the problem Kilbridge. Figure 2 shows the workstation workload is well-balanced and the solution is identified within 1s, which proves the efficiency of the proposed algorithm.

Table 2 shows the computational result comparison of the proposed algorithm with weighted algorithm

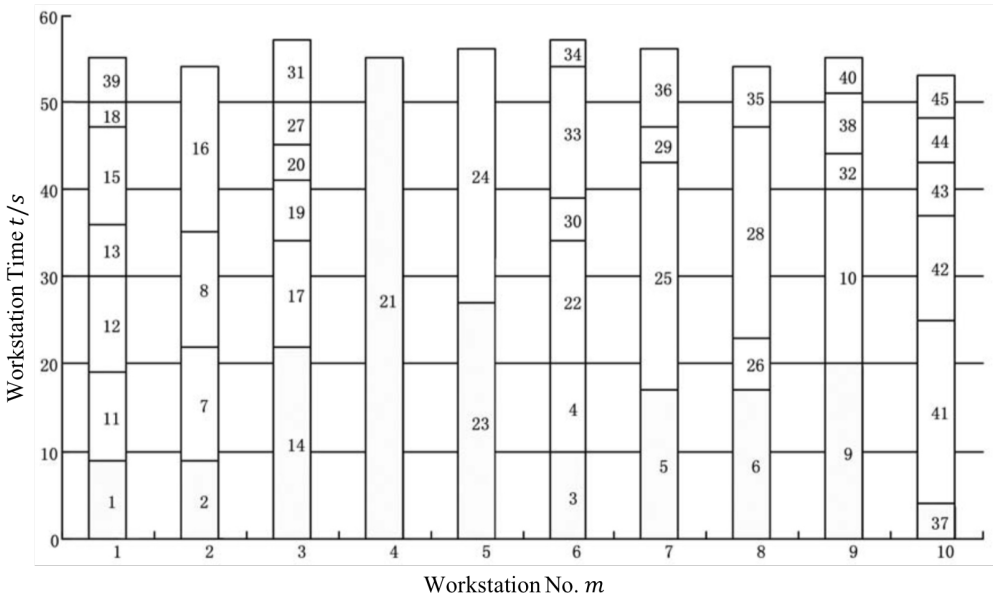

Figure 2: Task precedence graph of problem Kilbridge

and genetic algorithm (GA). It can be seen from the table that both the proposed ACO and GA can identify the solution with minimal number of workstations for all the 24 benchmarking problems and their overall performance is superior to the weighted algorithm. It is worth noting that the weighted algorithm uses a static precedence rule and its selection criterion is unique and deterministic; the proposed algorithm is stochastic in nature and it can fully utilize the information between ants and previous experience, which helps the algorithm jump out of local optima and therefore find the optimal solutions.

5 Conclusions

This paper proposes an adaptive ACO algorithm for the assembly line balancing problem. It encompasses the following characteristics: comprehensive considerations of three rules of utilization, exploration and random search to construct feasible assignment plans; comprehensive considerations of assembly line balancing rate as well as smoothing factor to evaluate balancing performance to increase differentiation capability of different solutions; adaptive modifications of global pheromone evaporation factor in the construction phase to help the algorithm jump out of local optima and improve its global search capability. Computational results validate the superior performance of the proposed algorithm.

This paper only considers single deterministic assembly balancing problem and the proposed algorithm can be used to solve more complex assembly balancing problems, like dynamic problem, stochastic problem, multi-objective problem and multi-model assembly line balancing problems.

Table 2: Computational result comparisons on benchmark problems

Problem	No. Tasks n	Cycle C	Time t_{sum}	Min. No. Workstation m_{min}	Adaptive ACO			Weight	GA
					m	L	I		
Jackson	11	10	46	5	5	92.0%	1.095	6	5
		13		4	4	88.5%	0.707	4	4
Heskiaoff	28	205	1024	5	5	99.9%	0.447	6	5
		256		4	4	100.0%	0	5	4
Buxey	29	30	324	12	12	90.0%	2.309	12	12
		41		8	8	98.8%	0.707	9	8
		54		7	7	85.7%	4.629	7	7
Sawyer	30	41	324	8	8	98.8%	0.707	9	8
		47		7	7	98.5%	2.390	8	7
Lutzl	32	1414	14140	11	11	90.9%	134.400	12	11
		1572		10	10	89.9%	134.270	11	10
		2020		8	8	87.5%	110.640	8	8
		2828		6	6	83.3%	449.620	6	6
Kilbridge	45	57	552	10	10	96.8%	2.240	11	10
		79		7	7	99.8%	0.377	8	7
		92		6	6	100.0%	0	7	6
		110		6	6	83.6%	18.876	6	6
		138		4	4	100.0%	0	5	4
		184		3	3	100.0%	0	4	3
Tonge	70	176	3510	21	21	95.0%	8.423	22	21
		364		10	10	96.4%	4.795	11	10
		410		9	9	95.1%	6.896	10	9
		468		8	8	93.8%	15.050	8	8
		527		7	7	95.1%	14.540	7	7

References

Joaquin Bautista and Jordi Pereira. Ant algorithms for assembly line balancing. In Marco Dorigo, Gianni Di Caro, and Michael Sampels, editors, *Ant Algorithms*, pages 65–75, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45724-4.

Joaqun Bautista and Jordi Pereira. Ant algorithms for a time and space constrained assembly line balancing problem. *European Journal of Operational Research*, 177(3):2016 – 2032, 2007. ISSN 0377-2217.

Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo. Distributed optimization by ant colonies. 01 1991.

Ozcan Kilincci. A petri net-based heuristic for simple assembly line balancing problem of type 2. *The International Journal of Advanced Manufacturing Technology*, 46(1):329–338, Jan 2010. ISSN 1433-3015.

Patrick R. McMullen and Peter Tarasewich. Using ant techniques to solve the assembly line balancing problem. *IIE Transactions*, 35(7):605–617, 2003.

Ugur O., Talip K., and Bilal T. A genetic algorithm for the stochastic mixed-model u-line balancing and sequencing problem. *International Journal of Production Research*, 49(6):1605–1626, 2011.

U. Ozcan and B. Toklu. A tabu search algorithm for two-sided assembly line balancing. *The International Journal of Advanced Manufacturing Technology*, 43(7):822, Sep 2008. ISSN 1433-3015.

Marc Peeters and Zeger Degraeve. An linear programming based lower bound for the simple assembly line balancing problem. *European Journal of Operational Research*, 168(3):716 – 731, 2006. ISSN 0377-2217. Balancing Assembly and Transfer lines.

S. G. 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, Jul 1999. ISSN 1433-3015.

M. E. Salveso. The assembly line balancing problem. *The Journal of Industrial Engineering*, 6(3):1–25, 1955.

Armin Scholl and Christian Becker. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168(3):666 – 693, 2006. ISSN 0377-2217. Balancing Assembly and Transfer lines.