An Improved Genetic Algorithm for Multi-objective Dynamic Scheduling Optimization

Kunlei Lian[1], Chaoyong Zhang[1], Liang Gao[1], and Chaoyang Zhang[1]

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

Abstract

Dynamic scheduling problem is a more complex NP-hard problem compared with static scheduling problem and in most cases it has multi-objective performance criteria. Based on study of the multi-objective dynamic scheduling problem, it emploies the rolling-horizon procedure and an iimproved genetic algorithm. In the rolling-horizon procedure, dynamic scheduling problem is decomposed ino a series of continual and static schedulig problems, multi-objective genetic algorithm is applied to each of tehse problems. In order to adapt to the complex manufacturing environment and sustain the stability of production, a human-compute collaborative scheduling procedure is presented for the implementation of the scheduling process. A scheduling prototype system is developed and tested on the improved job-shop benchmark instance, the simulation results validate the effectivenss of the proposed strategies.

Key words: Job shop scheduling problem; dynamic scheduling, genetic algorithm; rolling horizon

Traditional static scheduling models cannot be used in real-world manufacturing environment with dynamic, stochastic and multi-objective nature, and the concept of dynamic scheduling was first proposed by Jackson in 1957. The advent of dynamic scheduling models is due to the disturbed schedules inflicted by a series of stochastic impacting factors emcompassing the arrival of new orders, scrap and rework of certain parts, changed due date, machine malfunction or delayed raw material arrival. Rescheduling is therefore necessary in order to resume manufacturing process based on the current part status in the system.

Traditional integer programming methods can be hardly used in solving real-world dynamic scheduling problems. On the other hand, new advancemens in computer technologies open new opportunities in sovling dynamic scheduling problems using artificial intelligence, simulation, rolling-horizon rescheduling and meta-heuristics, which lays the foundations for practical usage of manufacturing scheduling models. Artificial

intelligence and expert systems can identify the best schedulig strategy by searching knowledge database based on current system status and predefined optimization objective. However, they suffer from poor adaptivity to new environment, high development cost and prolonged development cycle. Discrete system simulation method simulates real-world manufacturing environment by creating simulation models, from which general principles are difficult to find due to its experimental nature. Rolling horizon rescheudling was original proposed by Nelson in 1977 and has received numerous research attentions and applications in recent years. It divides the dynamic scheduling problem into continual scheduling sections and conducts on-line optimization for each section in order to find the individual optimal solution, which makes it applicable to complex dynamic manufacutring scheduling environments.

Genetic algorithm, one of the metaheuristic algorithms, has witnessed many applications in dynamic scheduling researches due to its simple operations, higher efficiency, better robustness, superior adaptability and intriguing searching capability based on individual fitness. This paper proposes a hybrid algorithmic framework of multi-objective genetic algorithm and rolling horizon rescheduling to solve the real-world dynamic scheduling problem considering delayed raw material arrival, part maching time and assembly time. It can also tackle the emergent insertion of parts and continuous arrival of scheduling jobs. This framework can be used in other manufacturing scenarios as well.

1 Assembly Line Balancing and Mathematical Model

ALBP refers to the assignment of finite job set to finite workstation set in order to maximize workstation utilization, minimize overall overload time and minimize balancing objective value, subject to processing constraints and workstation processing time satisfying 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:

$$\text{max.} \quad \lambda L - I \tag{1}$$

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

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

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

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

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

In this model, L is the balancing rate of an assembly line; I is the smoothing factor; λ is an 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.

2 An Adaptive ACO for the ALBP

2.1 Solution construction strategy

In order to use ACO to solve the ALBP, the processing jobs can be seen as nodes on a graph which will be traversed by ants, also the connections between jobs and workstations can be seen as arcs on the graph. The assignmnet of processing jobs to workstations can then be seen as an ant colony travels through the graph with the guidance of pheromone and heuristic information.

The solution construction is a key step in emplying ACO to solve ALBP and this paper constructs a feasible solution by gradually assigning processing jobs to corresponding workstations. To this end, we define the following notations:

- no-assigned task: a task that hasn't been assigned to any workstation yet
- available task: a task that hasn't yet been assigned but satisfy precedence constraints between tasks, also all of its preceeding tasks have been assigned
- assignable task: an available task that satisfies cycling constraints.

The feasible solution construction algorithm can then be described as follows:

1. open a workstation
2. identify the set of available tasks from all no-assigned tasks and the precedence constraints among tasks, if the resulting set is empty, go to step 7
3. identify the set of assignable tasks from available tasks and the cycling constraint
4. if the set of assinable tasks is empty, go to step 6
5. select a task from the set of assinable 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 assinable 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 corrresponding heuristic informaiton is computed by static precedence rules. In addition, the heuristic information of ACO consists of the maximal task completion time and maximal number of succedding 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 succedding 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:

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

where r is a random number chosen from (0, 1); r_1, r_2, r_3 is a user-defined parameter 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.

2.2 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 acculation 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 upating of pheromones and speeds up the algorithm's convergence rate.

2.3 Pheromone update strategy

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

- local pheromone trail update: 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 capacility 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 update: After all the ants finish traversal, on 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 searching iterations. This can increases 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.

2.4 Algorithm workflow

The overall algorithm works as follows

1. initialize algorithm parameters
2. open a 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 formual (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) repat 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. udpate 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.

3 Computational Experiments

Based on the above descriptios, we validated the performance of ACO on benchmark instances. The algorithm was 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 identified the parameters using multiple trails and the final paramters are set as follows: the number of ants (N_ANT_COUNT) = number of tasks (N_TASK_COUNT), the maximum number of iterataions $N_{C_{max}} = 50$, $\tau_0 = 1$, $\alpha = 1.0$, $\beta = 2.0$, $\rho_1 = 0.1$, $\rho_2(t_0) = 1$, $\rho_{2_{min}} = 0.5$, $r_1 = 0.6$, $r_2 = 0.3$, $r_3 = 0.1$.

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.

It can be seen from the computational results that our proposed algorithm was 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 was identified within 1s, which proved the efficiency of the proposed algorithm.

Table **??** shows the computational results of the proposed algorithm with weighted algorithm and genetic algorithm (GA). It can be seen from the table that both the proposed ACO and GA can identify the solution

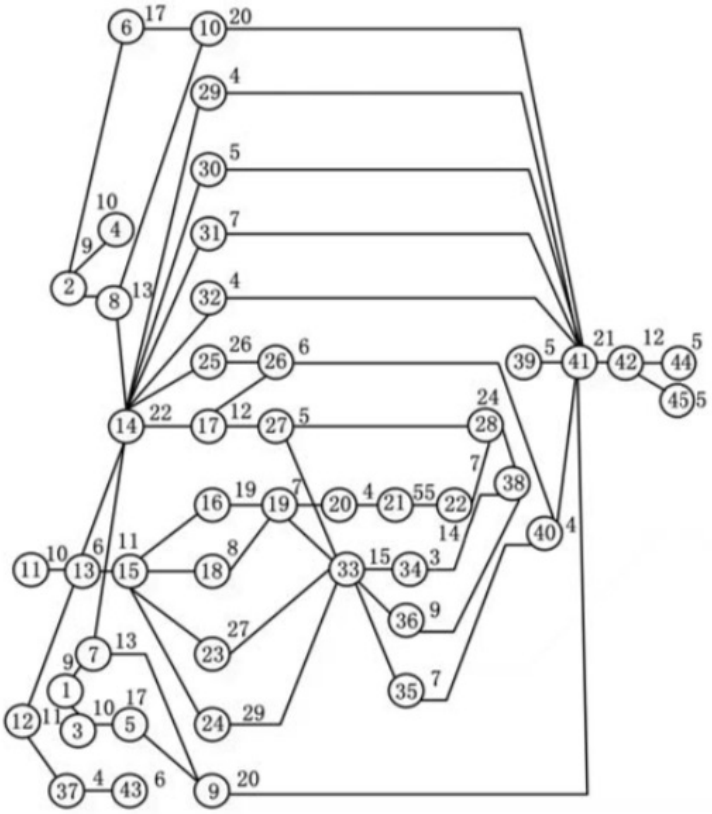

Figure 1: Task Precedence Graph of Problem Kilbridge

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

with minimal number of workstations for all the 24 benchmarking problems and their overall performance is superior than the weighted algorithm. It is worth noting that the weighted algorithm uses a static precendence 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.

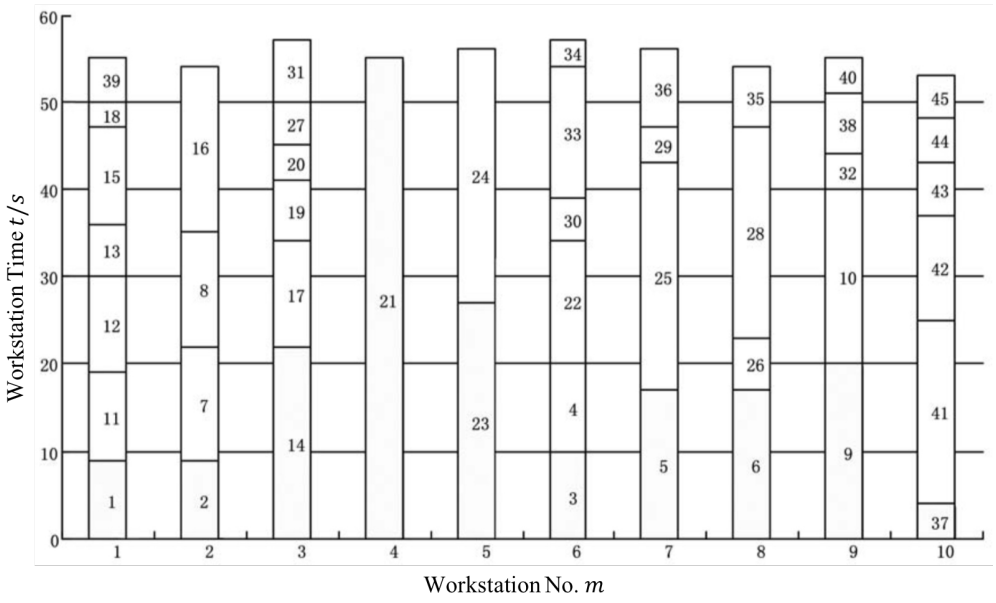

Figure 2: Task Precedence Graph of Problem Kilbridge

Table 2: Computational Result Comparisons on Benchmarking 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

4 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 contruct feasible assignment plans; comprehensive considerations of assembly line balancing rate as

well as smoothing factor to evaluate balancing performace 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. Computationsl results validated 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.

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. doi: https://doi.org/10.1016/j.ejor.2005.12.017. URL `http://www.sciencedirect.com/science/article/pii/S0377221705008490`.

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. doi: 10.1007/s00170-009-2082-z. URL `https://doi.org/10.1007/s00170-009-2082-z`.

Patrick R. McMullen and Peter Tarasewich. Using ant techniques to solve the assembly line balancing problem. *IIE Transactions*, 35(7):605–617, 2003. doi: 10.1080/07408170304354. URL `https://doi.org/10.1080/07408170304354`.

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. doi: 10.1007/s00170-008-1753-5. URL `https://doi.org/10.1007/s00170-008-1753-5`.

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. doi: https://doi.org/10.1016/j.ejor.2004.07.024. URL http://www.sciencedirect.com/science/article/pii/S0377221704004813. 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. doi: 10.1007/s001700050105. URL https://doi.org/10.1007/s001700050105.

M. E. Salveso. The assembly line balancing problem. *The Journal of Industrial Engineering*, 6(3):1–25, 1955. URL https://ci.nii.ac.jp/naid/10003095017/en/.

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. doi: https://doi.org/10.1016/j.ejor.2004.07.022. URL http://www.sciencedirect.com/science/article/pii/S0377221704004795. Balancing Assembly and Transfer lines.