International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:6, No:8, 2012

Unrelated Parallel Machines Scheduling Problem Using an Ant Colony Optimization Approach

Y. K. Lin, H. T. Hsieh, F. Y. Hsieh

***Abstract*—**Total weighted tardiness is a measure of customer satisfaction. Minimizing it represents satisfying the general requirement of on-time delivery. In this research, we consider an ant colony optimization (ACO) algorithm to solve the problem of scheduling unrelated parallel machines to minimize total weighted tardiness. The problem is NP-hard in the strong sense. Computational results show that the proposed ACO algorithm is giving promising results compared to other existing algorithms.

***Keywords*—**ant colony optimization, total weighted tardiness, unrelated parallel machines.

I. Introduction

PARALLEL machine scheduling problems have been studied by many researchers ([6]–[7]–[25]) in recent years. Parallel machine scheduling problems are important because most real world manufacturing workgroups have more than one machine. Parallel machine algorithms can be reduced for the single machine problems and also many job shop algorithms, such as shifting bottleneck heuristics, also called parallel machine algorithms. Though there is extensive literature on parallel machine scheduling problems, most of it is limited to situations in which the processing times or speed rates are the same across all machines. When machines are not identical to one another and cannot be completely correlated by simple rate adjustments, they are said to be unrelated parallel machines. Based on the complex hierarchy of deterministic scheduling ([7]–[25]) among the traditional classification of parallel machine environments, unrelated parallel machines are some of the most difficult problems to solve.

This research considers the problem of scheduling unrelated parallel machines to minimize total weighted tardiness (TWT). TWT is defined as $\sum w_j T_j$, where $T_j = max(C_j - d_j, 0)$, w_j is the weight of job j and d_j is the due date of job j. TWT is a measure of customer satisfaction. Minimizing it represents satisfying the general requirement of on-time delivery.

Y. K. Lin is with the Department of Industrial Engineering and Systems Management, Feng Chia University, Taichung, Taiwan (e-mail: yklin@mail.fcu.edu.tw).

H. T. Hsieh was with the Department of Industrial Engineering and Systems Management, Feng Chia University, Taichung, Taiwan (e-mail: m9803065@mail.fcu.edu.tw).

F. Y. Hsieh is with the Department of Industrial Engineering and Systems Management, Feng Chia University, Taichung, Taiwan (e-mail: m9902937@mail.fcu.edu.tw).

Following the three-field notation of [12], we refer to this problem as $R \| \sum w_j T_j$. The problem is NP-hard in the strong sense, since its special case with $m = 1$ is already NP-hard in the strong sense ([15]). Research related to scheduling parallel machines to minimize total weighted tardiness is limited. Based on shortest processing time (SPT) and earliest due date (EDD) principles, [22] proposed a heuristic (PSK) to minimize the mean tardiness for the single-machine sequencing problem ($1||\overline{T}_j$). [14] proposed a heuristic (KPM) for the $P||\overline{T}_j$ problem, based on extending the PSK heuristic from the single-machine tardiness problem to a parallel-machine setting. [1] presented a modified due date (MDD) algorithm for the $P||w_j T_j$ problem. [8] developed a heuristic algorithm based on tabu search to solve the problem of simultaneously selecting and scheduling parallel machines to minimize the sum of machine holding cost and job tardiness cost. [27] solved the $P||T_j$ problem using a branch-and-bound algorithm, and [5] proposed several efficient heuristic algorithms for the $P||T_j$ problem. According to [23], most unrelated parallel machine scheduling problems are NP-hard. [23] presented a survey of algorithms for single- and multi-objective unrelated parallel machine deterministic scheduling problems but indicated that unrelated parallel machine problems remain relatively unstudied. In particular, the study noted that there are a few solution approaches to minimize due-date-related criteria. [17] solved the $R||\sum w_j T_j$ problem using a branch-and-bound algorithm that can solve a problem size up to 4 machines and 18 jobs. The Apparent Tardiness Cost (ATC) rule proposed by [29], originally designed for the $1||\sum w_j T_j$ problem, can be used to solve the $R||\sum w_j T_j$ problem heuristically. [18] modified the ATC rule so that it can solve the $R||\sum w_j T_j$ problem. They proposed a heuristic, ATC-I, for the $R||\sum w_j T_j$ problem. Computational results show that the study's proposed ATC-I outperformed other existing heuristics (MDD, KPM, and ATC).

ACO is a constructive meta-heuristic that has been used to solve scheduling problems in recent years. [2] used an ACO algorithm to solve the single machine total tardiness problem ($1||\sum T_j$). [4] proposed an ACO algorithm that incorporates local search for the single machine total weighted tardiness problem ($1||\sum w_j T_j$).

[19] proposed an ACO algorithm that uses global pheromone information instead of using only local pheromone information for the $1||\sum w_j T_j$ problem. Similarly, [30] presented an ACO algorithm for the $1||\sum w_j T_j$ problem that can effectively improve the robustness of various simple constructive heuristics. Moreover, [13] proposed a fast ACO algorithm for the $1||\sum w_j T_j$ problem. [16] used an ACO algorithm for scheduling single machine tardiness problems with sequence-dependent setups ($1|s_{ij}|\sum w_j T_j$). [26] used an ACO algorithm to minimize makespan for identical parallel machines with sequence-dependent setup problems ($P|s_{ij}|C_{max}$). [31] and [20] both studied scheduling unrelated parallel machines to minimize total weighted tardiness using an ACO algorithm ($R||\sum w_j T_j$). [28] suggested an ACO algorithm to solve scheduling identical parallel machines to minimize total weighted tardiness ($P||\sum w_j T_j$). [21] used an ACO algorithm for scheduling jobs with incompatible families on parallel batch machines ($P|p\text{-}batch,incompatible|\sum w_j T_j$). [3] studied parallel machine scheduling problems with sequence-dependent setup times to minimize makespan ($P|s_{jk}|C_{max}$) using an ACO, a simulated annealing (SA), and a variable neighborhood search (VNS) hybrid algorithm.

In this research, we present an ACO algorithm for the $R||\sum w_j T_j$ problem.

II. Application of ACO algorithm

ACO ([9]–[10]–[11]) is a meta-heuristic for solving hard combinatorial optimization problems. It is inspired by the pheromone trail laying and following behavior of real ants, which uses pheromones as a communication medium. In a double bridge experiment conducted on ants, the result showed that after an initial transitory phase in which some oscillations can appear, eventually most of the ants choose the shortest path. The ACO algorithm imitates this optimization capability of selecting the shortest path from the ant colony system. The ACO algorithm is based on indirect communication within a colony of simple agents, called (artificial) ants, mediated by (artificial) pheromone trails.

The (artificial) ants in the ACO algorithm implement a randomized construction heuristic that probabilistically selects the solution components of the problem based on the pheromone trail and possibly available heuristic information, which is based on the input data of the problem to be solved. The (artificial) pheromone trails in the ACO algorithm serve as distributed, numerical information for the ants to probabilistically construct solutions to the problem being solved and to reflect their search experience. In this research, we apply the ACO algorithm to solve the problem of scheduling unrelated parallel machines to minimize total weighted tardiness. The following notation is used to define the problems.

m	The number of machines
n	The number of jobs
U	The set of unscheduled jobs
p_{ij}	The processing time of job j on machine i
w_j	The weight of job j
d_j	The due date of job j
t_i	Makespan of the scheduled jobs on machine i
$\tau_{ij}(t)$	Pheromone trail intensity of assigning job j on machine i in iteration t
η_{ij}	The heuristic desirability of assigning job j on machine i
α	Relative influence of pheromone information
β	Relative influence of heuristic information
ρ_{local}	Local pheromone evaporation rate ($0 < \rho_{local} < 1$)
ρ_{global}	Global pheromone evaporation rate ($0 < \rho_{global} < 1$)
q_{m0}	The user-specified number such that $0 \le q_{m0} \le 1$
q_{j0}	The user-specified number such that $0 \le q_{j0} \le 1$
$ANTS$	The number of ants in each ant colony
$ITER$	Total number of iterations to be run

The ACO algorithm

Step 1: Initialization. Set ACO parameters α, β, ρ_{local}, ρ_{global}, q_{m0}, q_{j0}, ITER, ANTS. Initialize, $TWT^* = \infty$, iter=1, ants=1, $U = \{1,...,n\}$, $t_i = 0, i = 1,2,...,m$.

Step 2: Machine selection. Let q_m be the random number generated from the uniform distribution [0,1] and q_{m0} be the user-specified number such that $0 \le q_{m0} \le 1$. Select the machine i^* from the following equation, $i^* = \begin{cases} min_{1\le i\le m}\, t_i, & if\ q_m \le q_{m0} \\ I, & otherwise \end{cases}$ where I is a probability distribution with $P_i = \dfrac{1/t_i}{\sum_{p=1}^{m} 1/t_p}$, $i = 1,2,...,m$.

Step 3: Job selection. Let q_j be the random number generated from the uniform distribution [0,1] and q_{j0} be the user-specified number such that $0 \le q_{j0} \le 1$. Select the job j^* from the following equation, $j^* = \begin{cases} arg\ max\{\tau_{i^*j}{}^{\alpha}(t)\cdot\eta_{i^*j}{}^{\beta}\}, j \in U & if\ q_j \le q_{j0} \\ J, & otherwise \end{cases}$ where

$$\eta_{i^*j} = \frac{w_j}{p_{i^*j}} exp\left(-\frac{w_j \, max(d_j - p_{i^*j} - t_{i^*}, 0)}{K\bar{p}_{i^*}}\right) \quad (1)$$

, and J is a probability distribution with $P_{i^*j} = \begin{cases} \frac{\tau_{i^*j}{}^{\alpha}(t) \cdot \eta_{i^*j}{}^{\beta}}{\sum_{l \in U} \tau_{i^*l}{}^{\alpha}(t) \cdot \eta_{i^*l}{}^{\beta}}, & if \ j \in U \\ 0, & otherwise \end{cases}$.

Step 4: Machine re-selection. For job j^*, find machine i^{**} such that $p_{i^{**}j^*}$ =min $\{p_{ij^*} \mid t_i + p_{ij^*} - d_{j^*} \le 0, 1 \le i \le m\}$. If i^{**} cannot be found from the above equation, conclude that there is no job that can be assigned without being tardy, then calculate i^{**} from $i^{**} = arg \ min_{1 \le i \le m}(t_i + p_{ij^*} - d_{j^*})$.

Step 5: Assign job j^* on machine i^{**}. Set $U = U \setminus j^*$ and $t_{i^{**}} = t_{i^{**}} + p_{i^{**}j^*}$. Update local pheromone for the selected machine i^{**} and job j^* as $\tau_{i^{**}j^*}(t) = (1 - \rho_{local})\tau_{i^{**}j^*}(t) + \rho_{local}\tau_0$ where $\tau_0 = 1/(ANTS \cdot TWT_{ATC\text{-}I})$, $TWT_{ATC\text{-}I}$ is the ATC-I ([18]) objective value.

Step 6: Repeat Steps 2-5 until all jobs are scheduled.

Step 7: Apply local search procedure to improve the given schedule.

Step 8: Find current TWT for the current solution. If $TWT < TWT^*$, then $TWT^* = TWT$. Update *ants*=*ants*+1. If $ants \le ANTS$, set $U = \{1,...,n\}$ and $t_i = 0, i = 1,2,...,m$ and go to Step 2. If $ants > ANTS$, go to Step 9.

Step 9: Update the global pheromone of the best solution found in the current iteration as $\tau_{ij}(t+1) = (1 - \rho_{global})\tau_{ij}(t) + \rho_{global}\Delta\tau_{ij}(t)$, where $\Delta\tau_{ij}(t) = 1/TWT^*$ and TWT^* is the best TWT value up to the current iteration. Set iter=iter+1. If $iter \le ITER$, then set *ants*=1, $U = \{1,...,n\}$ and $t_i = 0, i = 1,2,...,m$ and go to Step2. If *iter*> *ITER*, go to Step 10.

Step 10: Terminate the ACO algorithm.

In Step 2, when $q_m \le q_{m0}$, an ant selects the first available machine i^* that leads to the smallest makespan among the parallel machines (i.e,. $t_{i^*} = min_{1 \le i \le m} t_i$). When $q_m > q_{m0}$, then an ant randomly selects machine I from the probability distribution formed by the probabilities P_i.

In Step 3, once machine i^* is selected, the next decision of an ant is to select the job. Job selection considers both the heuristic information (η_{i^*j}) and pheromone trails ($\tau_{i^*j}(t)$). $\tau_{i^*j}(t)$ is the pheromone trail associated with the assignment of job j to machine i^*. The parameter t is used for the current iteration of the ACO algorithm. η_{i^*j} is the heuristic desirability of assigning job j to the selected machine i^*. η_{i^*j} is a modification of ATC index defined by Eq.(1). More precisely, we add w_j into the exponential term of the original ATC index. K is a scaling parameter that can be determined empirically, and $\bar{p}_{i^*} = \sum_{j=1}^{n} p_{i^*j}/n$ is the average job processing time of machine i^*. Hence, if K is very small, Eq. (1) reduces to the weighted minimal slack rule, (i.e., $w_j \, max(d_j - p_{i^*j} - t_{i^*}, 0)$), when there are no overdue jobs and otherwise to the weighted shortest processing time (WSPT) first rule for the overdue jobs. Parameters α and β determine the relative influence of the pheromone trails and the heuristic information. When $q_j \le q_{j0}$, an ant selects the job j^* that maximizes the value of $\{\tau_{i^*j}{}^{\alpha}(t) \cdot \eta_{i^*j}{}^{\beta}\}$. When $q_j > q_{j0}$, then an ant randomly selects job J from the probability distribution formed by the probabilities P_{i^*j}.

In Step 4, the ACO algorithm applies a greedy rule to find the fastest machine, i^{**}, from among the machines to which job j^* can be assigned without being tardy. If job j^* would be tardy on all machines, the ACO algorithm would choose the machine that would lead to the minimum total tardiness.

In Step 5, the ACO algorithm applies local pheromone updating. The intention of the pheromone update is to make solution components belonging to good solutions more desirable for ants operating in the following iterations.

In Step 7, the ACO algorithm uses local searches to further improve the schedule obtained so far. [3] introduced three local searches for the $P \mid S_{jk} \mid C_{max}$ problem: (1) LS1. Job swaps on one machine: randomly choose a machine *i*, and then randomly choose two jobs, j_1 and j_2 from machine *i*. Swap jobs j_1 and j_2. (2) LS2. Job swaps on different machines: randomly choose two machines i_1 and i_2, and then randomly choose two jobs, j_1 from machine i_1 and j_2 from machine i_2. Swap jobs j_1 and j_2. (3) LS3. Job insertion: randomly choose one job j_1 and one machine i_2, where j_1 does not belong to i_2. Randomly choose a valid position *r* in i_2. Transfer job j_1 to i_2 at position *r*. In our ACO algorithm, we always try to use LS1 first. If after LS1, no improvement is made, then another local search is used (*l* is incremented), and every time a new solution is found, the first local search is used (*l*=1).

In Step 9, global pheromone updating is applied. Global pheromone updating is intended to provide a greater amount of pheromone to the schedule with the best performance. Therefore, only the ant that found the best schedule up to the current iteration is permitted to deposit pheromone.

III. Computational results

In this section, we present several computational results on the performance of the proposed ACO algorithm. The ACO

algorithm was implemented in C++ and run on a computer with 2.5 GHz Pentium(R) Dual-Core CPU and 2G RAM. Processing times were generated from the uniform distribution [1,100]. The value of w_j for each j was chosen randomly from the uniform distribution [1,10]. Due dates were generated from the uniform distribution [$P(1 - T - R/2), P(1 - T + R/2)$], where $P = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}/m^2$, T is the average tardiness factor, and R is the relative range of due dates, as was done in [24]. We study cases with tight due dates since it is more challenging. Hence, due dates factor T was set to 0.8, and R was set to 0.4 and 0.8. We also used 4 machines with 20 jobs ($4m20n$) to represent small problem instances and used 10 machines with 100 jobs ($10m100n$) to represent large problem instances. For each combination of due date tightness and problem instance size, 20 problem instances were randomly generated. According to our extensive computational experimentation, ACO algorithm-related parameters were set to α=1, β=3, ρ_{local} =0.01, ρ_{global} =0.01, q_{m0} =0.9, q_{j0} =0.9, ANTS =20, and ITER=250.

A. Comparison of algorithms

[28] proposed an ACO algorithm (named ACO-SV) for the $P||\sum w_j T_j$ problem. Lin et al. (2011) proposed a genetic algorithm (GA) for the $R||\sum w_j T_j$ problem. We compare the proposed ACO algorithm with these two existing algorithms. The two algorithms ACO-SV and GA have been re-implemented in our work. For the small problem instances, we compare the average relative percentage deviations from the optimal solution obtained from the branch-and-bound algorithm proposed by [17] to the number of instances solved to optimal solution. For the large problem instances we compare the average relative percentage deviations from our proposed ACO algorithm and the number of instances solved to best heuristic solution. For each problem instance, K value for AU index in the ACO-SV algorithm and Eq. (1) in our proposed ACO algorithm was set to 0.6. Table 1 gives computational results for small problem instances, which show that the ACO algorithm outperformed the ACO-SV algorithm and the GA in terms of total weighted tardiness and number of instances solved to optimal solution.

TABLE I
THE PERFORMANCE OF THE ACO ALGORITHM (4*M*20*N*)

	T=0.8, R=0.4				T=0.8, R=0.8			
Instances	***Opt.***	***ACO***	***ACO-SV***	***GA***	***Opt.***	***ACO***	***ACO-SV***	***GA***
1	1118	**1118**	1287	**1118**	1336	**1336**	1530	1342
2	590	**590**	725	591	1398	**1398**	1547	1408
3	568	**568**	724	632	925	**925**	1203	968
4	1045	**1045**	1123	1060	690	**690**	824	770
5	2689	**2689**	2853	2751	2801	**2801**	2828	**2801**
6	927	**927**	999	987	258	**258**	356	**258**
7	1557	**1557**	1776	1665	855	**855**	1282	858
8	863	**863**	928	880	357	**357**	387	**357**
9	1188	**1188**	1243	**1188**	373	**373**	493	**373**
10	693	**693**	938	738	593	596	641	645
11	459	**459**	656	538	1428	**1428**	1675	1435
12	796	**796**	955	849	788	**788**	984	911
13	489	492	643	642	1437	**1437**	1645	**1437**
14	1468	**1468**	1706	1497	1679	**1679**	1786	**1679**
15	127	**127**	272	140	1023	**1023**	1332	1055
16	1528	**1528**	1593	1582	539	**539**	677	545
17	1375	**1375**	1730	1495	391	**391**	485	**391**
18	947	**947**	1118	970	479	**479**	595	511
19	433	**433**	596	451	1840	1842	2168	1879
20	624	**624**	754	650	457	**457**	522	**457**
Average	974.20	974.35	1130.95	1021.20	982.35	982.60	1148.00	1004.00
(Algo.-opt.)/opt.	-	0.0002	0.1609	0.0482	-	0.0003	0.1686	0.0220
Num. of opt. found	-	19	0	2	-	18	0	8

When the due dates were very tight ($T = 0.8, R = 0.4$), the ACO algorithm deviated 0.02% from the optimal solution, the ACO-SV algorithm deviated 16.09% from the optimal solution, and the GA deviated 3.51% from the optimal solution. Moreover, out of 20 problem instances, the ACO algorithm solved 19 problem instances to optimal solution, the ACO-SV solved 0 problem instances to optimal solution, and the GA solved 2 problem instances to optimal solution. When the due dates were tight ($T = 0.8, R = 0.8$), the ACO algorithm deviated 0.03% from the optimal solution, the ACO-SV algorithm deviated 16.86% from the optimal solution, and the GA deviated 2.2% from the optimal solution. Moreover, out of 20 problem instances, the ACO algorithm solved 18 problem instances to optimal solution, the ACO-SV solved 0 problem instances to optimal solution, and the GA solved 8 problem instances to optimal solution.

For large problem instances, we compared the average relative percentage deviations from our proposed ACO algorithm and the number of instances solved to best heuristic solution. The results from this experimentation are given in Table 2; they show that the proposed ACO algorithm outperformed other existing algorithms.

TABLE II
THE PERFORMANCE OF THE ACO ALGORITHM (10M100N)

	T=0.8, R=0.4			T=0.8, R=0.8		
Instances	***ACO***	***ACO-SV***	***GA***	***ACO***	***ACO-SV***	***GA***
1	**49**	501	84	**2673**	3322	2743
2	**521**	899	581	**3040**	3296	3180
3	**264**	1338	388	**2833**	3586	2980
4	**414**	936	698	**1406**	1667	1415
5	**392**	1581	592	**2652**	3232	2851
6	**281**	537	285	**2315**	2807	2509
7	**116**	641	191	**2994**	3314	3214
8	**159**	654	173	**1674**	2109	1753
9	**486**	933	523	**1810**	2021	1899
10	**682**	1077	708	**3578**	4079	3808
11	**455**	1177	477	**1534**	2049	1620
12	**327**	971	409	**1865**	2208	2093
13	**471**	1405	866	**3118**	3404	3205
14	**114**	443	271	3739	4291	**3681**
15	**266**	824	349	**1960**	2206	2015
16	**243**	1043	396	**2183**	3183	2434
17	**469**	785	574	**2849**	3659	2916
18	**686**	1639	811	**1857**	2475	1955
19	**127**	411	153	2725	3210	**2639**
20	**384**	936	392	**4358**	5114	4726
Average	345.30	936.55	446.05	2558.15	3061.60	2681.80
(Algo.-ACO)/ACO	-	1.7123	0.2918	-	0.1968	0.0483
Num. best heuristic	20	0	0	18	0	2

When the due dates were very tight ($T = 0.8, R = 0.4$), the ACO-SV algorithm deviated 171.23% from the ACO algorithm and the GA deviated 29.18% from the ACO algorithm. Moreover, the ACO found 20 best solutions (minimum TWT) out of 20 problem instances, both the ACO-SV algorithm and the GA found 0 best solutions out of 20 problem instances. When the due dates were tight ($T = 0.8, R = 0.8$), the ACO-SV algorithm deviated 19.68% from the ACO algorithm and the GA deviated 4.83% from the ACO algorithm. Moreover, the ACO found 18 best solutions (minimum TWT) out of 20 problem instances, the ACO-SV algorithm found 0 best solutions out of 20 problem instances, and the GA found 2 best solutions out of 20 problem instances.

IV. CONCLUSIONS

This research proposed an ACO algorithm for the problem of scheduling unrelated parallel machines to minimize total weighted tardiness. Computational results showed that the proposed ACO algorithm outperformed other existing algorithms (ACO-SV and GA) in terms of total weighted tardiness and the number of instances solved to best heuristic solution.

REFERENCES

[1] B. Alidaee, D. Rosa, "Scheduling parallel machines to minimize total weighted and unweighted tardiness," *Comput Oper Res* , vol. 24, pp. 775-788, August 1997.

[2] A. Bauer, B. Bullnheimer, R. F. Hartl, and C. Strauss, "An ant colony optimization approach for the single machine total tardiness problem, " Evolutionary Computation, CEC 99. Proceedings of the 1999 *Congress on Evolutionary Computation*. *IEEE Press*.

[3] J. Behnamian, M. Zandieh, and S. F. Ghomi, " Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm," *Expert Syst Appl*, vol. 36, pp. 9637-9644, August 2009.

[4] M. D. Besten, T. Stützle, and D. Dorigo, "Ant colony optimization for the total weighted tardiness problem," Lecture Notes in *Computer Science*, pp. 611-620, 2000.

[5] D. Biskup, J. Herrmann, and J. N. D. Gupta, "Scheduling identical parallel machines to minimize total tardiness," *Int J Prod Econ*, vol. 115, pp. 134-142, September 2008.

[6] J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt, and WeglarzJ, *Scheduling computer and manufacturing process*, Berlin, New York: Springer Verlag, 1996.

[7] P. Brucker, *Scheduling Algorithm (4th ed.)*. Berlin: Springer Verlag, 2004.

[8] D. Cao, M. Chen, and G. Wan, "Parallel machine selection and job scheduling to minimize machine cost and job tardiness," *Comput Oper Res*, vol. 32, pp. 1995-2012, 2005.

[9] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for distributed discrete optimization," *Artificial Life*, vol. 5, pp. 137–172, Spring 1999.

[10] M. Dorig and T. Stützle, *Ant colony optimization*. MIT Press, Cambridge, MA, 2004.

[11] M. Dorigo and T. Stützle, "Ant colony optimization: overview and recent advances," *Int Ser Oper Res Manage Sci*, vol. 146, pp. 227-263, 2010.

[12] R. Graham, E. Lawler, J. Lenstra, and K. A. Rinnooy, "Optimization and approximation in deterministric sequencing and scheduling: A survey," *Ann Discrete Math*, vol. 5, pp. 287-326, 1979.

[13] O. Holthaus and C. Rajendran, " A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs," *Journal of the Operational Research Society*, vol. 56, pp. 947-953, August 2005.

[14] C. Koulamas, "Decomposition and hybrid simulated annealing heuristics for the parallel-machine total tardiness problem, " *Naval Research Logistics*, vol. 44, pp. 109-125, February 1997.

[15] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Bricker, "Complexity of machine scheduling problems," *Ann Discrete Math*, vol. 1, pp. 343-362, 1977.

[16] C. J. Liao and H. C. Juan, "An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups," *Computers and Operations Research*, vol. 34, pp. 1899-1909, August 2007.

[17] C. F. Liaw, Y. K. Lin, C. Y. Cheng, and M. C. Chen, "Scheduling unrelated parallel machines to minimize total weighted tardiness, " *Comput Oper Res*, vol. 30, pp. 1777-1789, January 2003.

[18] Y. K. Lin, M. E. Pfund, and J. W. Fowler, "Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems," *Comput Oper Res*, vol. 38, pp. 901-916, June 2011.

[19] D. Merkle and M. Middendorf, "Ant colony optimization with global pheromone evaluation for scheduling a single machine," *Applied Intelligence*, vol. 18, pp. 105-111, 2003.

[20] L. Mönch, "Heuristics to minimize total weighted tardiness of jobs on unrelated parallel machines," Proceedings of the *4th IEEE Conference on Automation Science and Engineering*, 2008, pp. 572-577.

[21] L. Mönch and C. Almeder, "Ant colony optimization approaches for scheduling jobs with incompatible families on parallel batch machines," Dublin, Ireland, *Multidisciplinary International Conference on Scheduling, Theory and Applications*, 2009, pp. 105-114.

[22] S. S. Panwalkar, M. L. Smith, and C. P. Koulamas, "A heuristic for the single machine tardiness problem," *Eur J Oper Res*, vol. 70, pp.304-310, November 1993.

[23] M. Pfund, J. W. Fowler, and J. N. D. Gupta, "A survey of algorithm for single and multi-objective unrelated parallel-machine deterministic scheduling problems," *Journal of the Chinese Institute of Industrial Engineers* vol. 21, pp. 230-241, 2004.

[24] C. N. Potts and L. N. Van Wassenhove, "A decomposition algorithm for the single machine total tardiness problem," *Operations Research Letters*, vol. 1, pp. 177-181, 1982.

[25] M. Pinedo, *Scheduling Theory, Algorithms, and Systems*, 3rd ed., Prentice Hall, 2008.

[26] S. S. Sankar, S. Ponnambalam, V. Rathinavel, and M. Visveshvaren, " Scheduling in parallel machine shop: an ant colony optimization approach," *Industrial Technology*, 2005, pp. 276-280. Hong Kong: IEEE.

[27] S. O. Shim and Y. D. Kim, "Scheduling on parallel identical machines to minimize total tardiness," *Eur J Oper Res*, vol. 177, pp. 135-146, February 2007.

[28] N. R. Srinivasa Raghavan and M. Venkataramana, "Parallel processor scheduling for minimizing total weighted tardiness using ant colony optimization," *Int J Adv Manuf Tech*, vol. 41, pp. 986–996, May 2009.

[29] A. P. J. Vepsalainen and T. E. Morton, "Priority rules and lead time estimation for job shop scheduling with weighted tardiness costs," *Manage Sci*, vol. 33, pp. 1035-1047, August 1987.

[30] K. C. Ying and C. J. Liao, "An ant colony system approach for scheduling problems," *Production Planning and Control: The Management of Operations*, vol.14, pp. 68-75, November 2003.

[31] H. Zhou, Z. Li, and X. Wu, "Scheduling unrelated parallel machine to minimize total weighted tardiness using ant colony optimization," Proceedings of the *IEEE International Conference on Automation and Logistics*, 2007, pp. 132-136