Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)

An Ant Colony Optimization Approach to the Minimum Tool Switching Instant Problem in Flexible Manufacturing System

Abdullah Konak and Sadan Kulturel-Konak

Abstract— **Efficient tool management is very important for the productivity in flexible manufacturing systems. This paper proposes an Ant Colony Approach to minimize the number of tool switching instants in flexible manufacturing systems for the first time. The proposed approach is compared to optimal results from the literature, and very promising results are reported.**

I. Introduction

In flexible manufacturing systems (FMS), parts are processed on the Computer Numerical Control (CNC) machines with tool magazines on which the required tools are loaded. In case the total number of tools required by all part types is larger than the tool magazine capacity, tool loading or switching between the processing of the part types become inevitable. Tool loading or switching usually consumes time and therefore may delay the planned production. Hence, a successful tool management is very important for high productivity in FMS. The importance of the tool switching problem has been recognized in the automated manufacturing literature over the last two decades. The relevant research has considered two objectives: minimizing the number of tool switches and minimizing the number of tool switching instants.

Minimizing the number of tool switches is relevant to the case where the tool switching time is significant compared to the processing times. Crama et al. [1] showed that minimizing the number of tool switches problem is strongly NP-hard. Tang and Denardo [2] showed that the problem is polynomially solvable when the part type sequence is given. Several procedures for solving this problem were proposed in the literature [3], [4], [5].

Minimizing the number of tool switching instants is appropriate when the automatic tool interchanging device can switch a number of tools simultaneously or when the tool switch time is independent of the number of the tool switches. The problem is particularly important when the tool loading times are long; therefore, the efficient loading of the tool magazines becomes crucial in minimizing the number of tool switching instants. Tang and Denardo [6] and Denizel [7] studied the minimum number of the tool switching instants problem. Tang and Denardo [6] showed that the problem generalizes the classical bin packing problem; therefore, it is NP-hard. They also showed that the problem can be formulated as a part type grouping problem and proposed a branch and bound procedure to find the optimal grouping. Denizel [7] proposed a Lagrangean decomposition based lower bounding procedure and uses the lower bound in a branch and bound algorithm. However, this approach is limited to the problems up to 30 part types.

In this paper, we consider the minimum number of tool switching instants problem (MTSIP) and developed an Ant Colony Optimization (ACO) approach to solve large sized problems with practical importance. An ACO approach has not been previously applied to MTSIP. The Integer Programming (IP) formulation of MTSIP is given as follows:

<u>Decision Variables and Parameters</u>

$$x_{ij} = \begin{cases} 1 & \text{part type } j \text{ is scheduled at instant } i, \\ 0 & \text{otherwise.} \end{cases}$$

$$y_{il} = \begin{cases} 1 & \text{if tool } l \text{ is assigned to instant } i, \\ 0 & \text{otherwise.} \end{cases}$$

$$b_i = \begin{cases} 1 & \text{instant } i \text{ is considered,} \\ 0 & \text{otherwise.} \end{cases}$$

$C =$ the capacity of the tool magazine

<u>Model:</u>

$$\text{Min } z = \sum_{i=1}^{N} b_i$$

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad \forall j \qquad \text{(C1)}$$

$$\sum_{j \in s_l} x_{ij} \le N y_{il} \qquad \forall i,l \qquad \text{(C2)}$$

$$\sum_{l=1}^{M} y_{il} \le C\, b_i \qquad \forall i \qquad \text{(C3)}$$

$$x_{ij}, y_{il}, b_i \in \{0,1\}$$

Constraint (C1) ensures that each part type *j* is assigned to exactly one instant. Given s_l is the set of parts requiring tool type *l*, constraint (C2) states that part type *j* can be scheduled in instant *i* only if all its required tools are on the tool magazine. The tool magazine capacity constraint is given in (C3).

MTSIP is a special version of the bin packing problem

Manuscript received October 24, 2006

Abdullah Konak is with the Pennsylvania State University Berks, Information Sciences and Technology, Reading, PA 19610 USA (corresponding author, phone: 610-396-6310; fax: 610-396-6024; e-mail: konak@psu.edu).

Sadan Kulturel-Konak is with the Pennsylvania State University Berks, Management Information Systems, Reading, PA 19610 USA (e-mail: sadan@psu.edu).

with sharing, i.e., parts may require common tools. If the part types do not share any common tool, then the problem reduces to the bin packing problem where the objective is to minimize the number of the bins. It should be noted that although ACO has previously been applied to the bin packing problem [8] and [9], our approach is different due to this tool sharing property of the problem.

II. Ant Colony Optimization

ACO was first proposed by Dorigo et al. [10, 11] as a new approach to the traveling sales-person problem (TSP) and inspired by the behavior of real ants while searching for food. Initially, real ants randomly wander to search for food. When an ant finds food, she will leave a chemical message called pheromone on the way to her nest so that other ants are invited to follow this chemical trail instead of wandering randomly. Over the time, however, the pheromone trail will loose its attraction as it constantly evaporates. Therefore, a shorter route from the food source to the nest is likely to retain a higher level of pheromone than a longer route does, in turn attracting more ants to follow. Additionally, ants following a pheromone trail will reinforce it by laying their pheromone trails as well. Over the time, a shorter route to the food source will attract more ants than a longer route. However, some other ants keep wandering randomly and may discover shorter routes to the food source, establishing a new stronger pheromone trail.

Dorigo and Gambardella [11] proposed a new search algorithm to solve the TSP inspired by the way ants search for food. Similar to Genetic Algorithms (GA), ACO operates on a set of solutions (population) in parallel. The population mimics a real ant colony, and each member ant represents a solution. In ACO, new solutions are randomly constructed from scratch at each cycle unlike Simulated Annealing (SA), Tabu Search (TS), and GA where new solutions are generated from existing ones using local and global search operators. Dorigo and Gambardella [11] defined an artificial pheromone $\tau(i,j)$ for each city pair i and j. The value of $\tau(i,j)$ represents the relative desirability that city j succeeds city i in a feasible TSP tour. Each ant constructs a solution by starting from a random city and randomly traversing one city after another until all cities are visited. The probability that an ant currently in city i visits unvisited city j next in the tour is given as a function of the pheromone as follows:

$$p(i,j)=\begin{cases}\dfrac{\tau(i,j)^{\alpha}\,\eta(i,j)^{\beta}}{\sum_{k\in V}\tau(i,k)^{\alpha}\,\eta(i,k)^{\beta}} & \text{if } j\in V\\[2ex] 0 & \text{otherwise}\end{cases} \tag{1}$$

where V is the set of the cities that have not yet been visited by the ant, $\eta(i,j)$ is a problem specific information (i.e., the distance between cities i and j in this case), α and β are parameters to set the relative importance of the pheromone trail information and the problem specific information.

After all ants in the population have constructed their solutions, the pheromone trail is updated as follows:

$$\tau(i,j)=\rho\times\tau(i,j)+\sum_{k\in P}\frac{\omega(i,j,k)}{z(k)} \tag{2}$$

where $z(k)$ is the total distance of the tour created by the k^{th} ant, and $\omega(i,j,k)=1$ if city j succeeds city i in solution k, otherwise $\omega(i,j,k)=0$. The first part of (2) represents the pheromone evaporation where ρ is the evaporation parameter between 0 and 1. The second part represents pheromone deposition by the ants. Thereby, if city j frequently succeeds city i in solutions with short tours, pheromone $\tau(i,j)$ becomes stronger and stronger, increasing the probability that city j succeeds city i in solutions that will be generated in the future cycles.

Many researchers have proposed improvements over the original ACO algorithm which briefly introduced above. Surveys of recent research efforts and developments in ACO are given in [12], [13], and [14]. Since [11], ACO have been applied to variety of optimization problems including graph colouring [8], bin packing/stock cutting [9], [15], [16], [17] scheduling [18], [19], [20], [21], [22], [23], redundancy allocation problem [24], and network routing [25].

III. Applying ACO to the Problem

An ACO implementation involves several steps: defining meaningful pheromone trail τ, identifying a proper problem specific information η, and defining an effective mechanism to update the pheromone trails. In the following sections, we describe the details of our implementation.

A. Pheromone trail definition

The definition of pheromone trails is very important for a successful ACO implementation. In ACO, pheromone trails are the communication channels through which ants exchange information about the solution space [26]. Therefore, a meaningful medium of communication plays a major role in the utilization of collective knowledge of ants to investigate the promising regions of the solution space. MTSIP is a variation of the bin packing problem. We defined $\tau(i,j)$ as the favorability of processing part types i and j in the same tool switching instant. This definition is similar to the pheromone trail definitions in the previous ACO approaches to the bin packing problem [8], [9]. By this definition, the pheromone matrix is symmetric, i.e., $\tau(i,j)=\tau(j,i)$.

B. Building a feasible solution

Let $\prod=\{S_1, S_2,\ldots, S_m\}$ represent a solution with m tool switching instants where S_i is the set of parts in the i^{th} instant. In other words, S_i represents a set of parts that are processed on a single CNC machine without requiring a tool switch once all required tools are loaded to the magazine. Each ant starts with an empty solution, $\prod=\{S_1\}$ and $S_1=\varnothing$. Then, parts are randomly assigned to S_1 one by one until the full tool capacity of the magazine. If no more parts can be assigned to the current instant, then a new instant is created, i.e., $\prod=\{S_1, S_2\}$ and $S_2=\varnothing$. This process continues until all parts have been assigned. Let S_m represent the current instant and $A(S_m)$ be the set of admissible parts that can be assigned to S_m. Admissible set $A(S_m)$ includes the parts that have not been assigned to an instant yet, and that would not violate the magazine capacity constraint when they are assigned to S_m. The probability of assigning part $i\in A(S_m)$ to S_m is given by:

$$p(i,S_m)=\begin{cases}\dfrac{\tau(i)\eta(i)^{\beta}}{\sum_{j\in S_m}\tau(j)\eta(j)^{\beta}} & \text{if } i\in A(S_m)\\ 0 & \text{otherwise}\end{cases}\qquad(3)$$

where $\tau(i)$ is the total pheromone trail of part i and $\eta(i)$ is the problem specific information. Let s(i) represent the set of tools required by part i and $|s(i)|$ be its cardinality. Total pheromone trail $\tau(i)$ depends on whether current instant S_m is empty or not as given in (4). If S_m is not an empty set, $\tau(i)$ is defined as the sum of the pheromone trails between part i and the parts already assigned in S_m. If S_m is an empty set, i.e., a new instant has just been created, $\tau(i)$ equals to $|s(i)|$, the number of the tool types required by part i. Therefore, while assigning parts to an empty instant, parts with a higher number of tool requirements are preferred with respect to the ones with a less number of tool requirements.

$$\tau(i)=\begin{cases}\sum_{j\in S_m}\tau(i,j) & \text{if } S_m\neq\varnothing\\ |s(i)| & \text{otherwise}\end{cases}\qquad(4)$$

We defined problem specific information $\eta(i)$ as a function of the residual capacity of the magazine that would be available after assigning admissible part i to S_m as follows:

$$\eta(i)=C+1-|s(S_m\cup i)|\qquad(5)$$

where $s(S_m\cup i)$ is the set of tools required by the set of parts in S_m and part i together. While assigning new parts to the current instant S_m, this approach favors parts that would require less number of new tools in addition to the currently loaded ones on the magazine over the ones that would require several new tools.

C. Initializing and updating the pheromone trail

As mentioned earlier, pheromone trail $\tau(i,j)$ represents the favorability of processing part i and j in the same tool switching instant. The pheromone trails are initialized according to this definition as given in (6). The default initial value of $\tau(i,j)$ is set to the number of parts, N. If two parts i and j share common tools, then $\tau(i,j)$ is increased by the number of the common tools that they share.

$$\tau(i,j)=N+|s(i)\cap s(j)|\qquad(6)$$

At the end of each cycle, the pheromone trails are updated. There are several approaches proposed in the literature to update pheromone trails (see [14] for a comprehensive summary). An important concern while updating pheromone trails is to prevent the stagnation behavior, i.e., the situation in which all ants create the same solution [10]. In the original ACO algorithm, all ants are allowed to deposit pheromone proportional to their fitness. This approach usually results in a slow convergence. Another strategy is the elitist approach where only the best ants found so far in the search (i.e., the global best) are allowed to deposit pheromone. In the MAX-MIN Ant System [27], only one ant, either the global best or the best ant in a cycle, is allowed to update the pheromone trails. Using the global best promotes exploitation while using the cycle best promotes exploration properties of the search. Switching between the global best and cycle best allows balancing exploration versus exploitation [9]. In addition, to minimize the stagnation behavior, upper and lower bounds are imposed on the pheromone trails. The disadvantage of the MAX-MIN Ant System is that it introduces three additional algorithm parameters.

We propose a different approach to update pheromone trails than previously reported ones because of the unique properties of the problem. MTSIP has a very flat objective function space. Therefore, there is a high probability that multiple solutions generated during a cycle have the same objective function value. In most cycles, we observed several solutions with the same objective function value. Let $\Pi_B(t)$ be the set of unique solutions with the best objective function value in cycle t. Note that the same solution may be generated by multiple ants in a cycle - especially this occurs as the pheromone trails converge toward the end of the search. Set $\Pi_B(t)$ includes only one copy of them. The pheromone trails are updated as follows:

$$\tau(i,j)=\rho\times\tau(i,j)+\sum_{\Pi\in\Pi_B(t)}\omega(i,j,\Pi)\qquad(7)$$

where ρ is the evaporation parameter indicating the rate of the decay of the pheromone trail, and $\omega(i,j,\Pi)=1$ if parts i and j are in the same instant of solution $\prod$, and $\omega(i,j,\Pi)=0$ otherwise. Instead of a single ant, allowing multiple peer ants to update the pheromone trails improves both exploration and exploitation of the algorithm. If a part pair i and j appears together in the same instant of several best solutions, $\tau(i,j)$ is increased aggressively, resulting in exploitation around solution schemas in which parts i and j are together. Considering multiple solutions while updating the pheromone trail helps exploration as multiple solution structure can be searched in parallel. This also prevents the premature convergence of the algorithm.

D. Overall Algorithm

The overall steps of our algorithm are given below. It should be noted that our approach does not employ any advanced features to improve the performance of ACO. One of the popular ways to improve the performance of an ACO implementation is local search [28]. ACO and local search can be hybridized in various ways. For example, solutions found by ACO can be run through a local search to improve them further, and then the improved solutions are used to update the pheromone trails, or the pheromone trails are randomly modified [14]. However, we did not incorporate any local search procedure because of three reasons: (*i*) to design a simple algorithm with a minimum possible number of parameters to be set by the user, (*ii*) local search procedures are usually not effective for problems with flat objective spaces, and finally (*iii*) we aim to investigate the performance of a pure ACO approach to MTSIP. In addition, our test results are very promising without using a local search procedure.

```
t←0
initialize $\tau(i, j)$
While ($t \leq t_{\max}$) do
    Randomly create $n_{\text{pop}}$ solutions using probabilities in (3)
    Sort the population
    Identify and delete duplicate solutions
    Update best-solution-so-far
    Update $\tau(i, j)$ using the best solutions of the cycle
    t←t+1
End
Return best-solution-so-far
```

IV. Computational Experiments

Ten different problem sets with 30 problems in each were used to test the effectiveness of the proposed ACO approach. The parameters of these problem sets are given in Table I. Problem sets TDI, TDII, and TDIII were defined by Tang and Denardo [6]. We randomly generated 30 random problems for each set as described in [6]. Test problems set DI to DVI were used by Denizel [7]. Denizel [7] also defined 30 random problems for each problem set and reported the ranges for the optimal solutions. The input data of these problems were provided us by the author. Therefore, we were able to compare the performance of our approach with respect to optimal solutions. The ACO computer code was coded in the C programming language and run using Linux PC with 2.8 GHz CPU and 1.5GB memory.

The results found for both test sets are also given in Table I. The parameters of the proposed ACO in all runs were set as follows: n_{pop}=50, $t_{\max}$=1000, ρ=0.95, and β=1. For each problem instance, ten random replications were performed. In the table, we reported the minimum and maximum number of tool switching instants found in each instance over ten replications as [min, max]. If the same result found in all random replications of a problem, this result is given without using brackets. We also provided the minimum and maximum (i.e., the range) and the average number of tool switching instants found for each problem set. Finally, we reported the average CPU seconds.

Unfortunately for the Tang and Denardo problem sets, there is no published result available to compare our findings. However, we were able to solve the IP formulation of the problem for the first set (TDI) using CPLEX v9.0. We reported the associated optimal results of this set under the column titled "Optimal." The optimal solutions for the problems of sets TDII and TDIII could not be obtained within reasonable CPU times. Denizel [7] did not report the optimal solution for each individual problem of DI-DVI, but provided the range and average value over the 30 problems in each set. We also reported ACO solutions in the same format in Table I.

When the final results were analyzed, one of the strengths of the proposed ACO approach appeared to be its robustness. As seen in Table I, for all cases except problem #19 of set TDIII, the ACO approach found the same objective function values in all ten random replications. When the final solutions were analyzed, it was realized that the final solutions were slightly different from each other with the same objective function. As discussed earlier, this should be expected in similar problems with flat objective function surfaces. In fact, this is one of the motivations for using an ACO approach to solve the problem as apposed to a metaheuristic based on local search strategy such as TS.

In terms of the solution quality, the ACO approach also performed superbly. The ACO approach found an optimal solution for all 30 problems of set TDI. For problem sets DI, DII, DIII, DV, and DVI, the ACO approach found the same optimal range and the average optimal solution quality over 30 problems. For set DIV, the final results were very close to the previously reported optimal results.

V. Conclusions

In this paper, for the first time an ACO approach was developed for the minimum number of tool switching instant problem. The results showed that the proposed ACO approach is very promising. For the large majority of the test problems, the proposed ACO approach found the optimal solutions. It would be an interesting further research to compare the proposed ACO approach with the other metaheuristic approach on larger sized problems.

TABLE I
TEST PROBLEM DATA AND RESULTS

Parameter	TDI		TDII	TDIII	DI	DII	DIII	DIV	DV	DVI
N	10		20	30	20	30	30	25	25	30
L	10		15	25	15	20	20	30	35	30
C	4		8	10	8	10	12	16	18	19
Problem #	**TDI**	**TDI Optimal**	**TDII**	**TDIII**	**DI**	**DII**	**DIII**	**DIV**	**DV**	**DVI**
1	6	6	5	12	5	9	6	5	6	5
2	3	3	7	15	6	7	5	5	6	4
3	3	3	5	9	6	7	5	5	6	4
4	4	4	13	11	6	7	5	5	6	5
5	4	4	7	13	7	8	5	5	5	4
6	4	4	7	13	7	8	5	5	5	4
7	5	5	8	12	5	7	5	5	5	4
8	4	4	7	7	4	10	7	6	6	4
9	4	4	3	16	6	7	5	5	6	4
10	4	4	6	15	7	8	6	4	5	4
11	3	3	9	14	5	8	5	5	5	4
12	4	4	7	16	6	7	5	5	6	4
13	4	4	10	15	6	8	5	5	5	4
14	3	3	7	9	5	7	5	6	6	5
15	4	4	11	9	7	8	5	6	6	4
16	4	4	5	15	6	7	5	5	6	4
17	4	4	8	10	4	8	6	5	5	4
18	3	3	9	14	5	8	5	5	5	4
19	3	3	6	[11,12]	9	6	4	7	7	5
20	4	4	7	9	7	8	5	6	6	5
21	4	4	2	13	7	6	5	5	6	4
22	5	5	7	13	6	7	5	7	7	5
23	4	4	7	11	7	7	5	5	6	4
24	4	4	5	9	7	7	5	6	6	5
25	5	5	7	14	5	7	5	5	6	4
26	4	4	9	11	5	7	5	5	6	4
27	5	5	10	13	6	6	4	6	7	5
28	5	5	8	12	5	8	5	5	6	4
29	4	4	6	14	5	9	6	6	6	5
30	4	4	6	8	6	6	4	6	6	4
Range	**[3, 6]**	**[3,6]**	**[2, 13]**	**[7, 16]**	**[4, 9]**	**[6, 10]**	**[4, 7]**	**[4, 7]**	**[5, 7]**	**[4, 5]**
Average	**4.03**	**4.03**	**7.13**	**12.11**	**5.93**	**7.43**	**5.1**	**5.38**	**5.83**	**4.3**
CPU	**3.38**	**NA**	**9.05**	**21.93**	**9.35**	**19.05**	**18.62**	**15.98**	**17.72**	**21.88**
Optimal Range from [7]→					**[4, 9]**	**[6, 10]**	**[4, 7]**	**[4, 6]**	**[5, 7]**	**[4, 5]**
The average of the optimal solutions [7]→					**5.9**	**7.4**	**5.1**	**4.6**	**5.8**	**4.3**

REFERENCES

[1] Y. Crama, A. W. J. Kolen, A. G. Oerlemans, and F. C. R. Spieksma, “Minimizing the number of tool switches on a flexible machine,” *International Journal of Flexible Manufacturing Systems*, vol. 6, pp. 33-54, 1994.

[2] C. S. Tang and E. V. Denardo, “Models arising from a flexible manufacturing machine. I. Minimization of the number of tool switches,” *Operations Research*, vol. 36, pp. 767-77, 1988.

[3] J. F. Bard, “A heuristic for minimizing the number of tool switches on a flexible machine,” *IIE Transactions*, vol. 20, pp. 382-91, 1988.

[4] G. Laporte, J. J. Salazar-Gonzalez, and F. Semet, “Exact algorithms for the job sequencing and tool switching problem,” *IIE Transactions*, vol. 36, pp. 37-45, 2004.

[5] M. A. Al-Fawzan and K. S. Al-Sultan, “A tabu search based algorithm for minimizing the number of tool switches on a flexible machine,” *Computers and Industrial Engineering*, vol. 44, pp. 35-47, 2003.
[6] C. S. Tang and E. V. Denardo, “Models arising from a flexible manufacturing machine. II. minimization of the number of switching instants,” *Operations Research*, vol. 36, pp. 778-84, 1988.
[7] M. Denizel, “Minimization of the number of tool magazine setups on automated machines: a Lagrangean decomposition approach,” *Operations Research*, vol. 51, pp. 309-20, 2003.
[8] D. Costa and A. Hertz, “Ants can colour graphs,” *Journal of the Operational Research Society*, vol. 48, pp. 295-305, 1997.
[9] J. Levine and F. Ducatelle, “Ant colony optimization and local search for bin packing and cutting stock problem,” *Journal of the Operational Research Society*, vol. 55, pp. 705-16, 2004.
[10] M. Dorigo, V. Maniezzo, and A. Colorni, “Ant system: optimization by a colony of cooperating agents,” *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 26, pp. 29-41, 1996.
[11] M. Dorigo and L. M. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 53-66, 1997.
[12] M. Dorigo and C. Blum, “Ant colony optimization theory: A survey,” *Theoretical Computer Science*, vol. 344, pp. 243-78, 2005.
[13] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo, “Model-based search for combinatorial optimization: a critical survey,” *Annals of Operations Research*, vol. 131, pp. 373-95, 2004.
[14] S. D. Shtovba, “Ant algorithms: theory and applications,” *Programming and Computer Software*, vol. 31, pp. 167-78, 2005.
[15] G. Bilchev, “Evolutionary metaphors for the bin packing problem,” San Diego, CA, USA, 1996.
[16] B. Brugger, K. F. Doerner, R. F. Haiti, and M. Reimann, “AntPacking - an ant colony optimization approach for the one-dimensional bin packing problem,” Coimbra, Portugal, 2004.
[17] A. Cuesta-Canada, L. Garrido, and H. Terashima-Marin, “Building hyper-heuristics through ant colony optimization for the 2D bin packing problem,” Melbourne, Vic., Australia, 2005.
[18] C. Rajendran and Y. Gajpal, “An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops,” *International Journal of Production Economics*, vol. 101, pp. 259-72, 2006.
[19] C. Rajendran and H. Ziegler, “Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs,” *European Journal of Operational Research*, vol. 155, pp. 426-38, 2004.
[20] C. Rajendran and H. Ziegler, “Two ant-colony algorithms for minimizing total flowtime in permutation flowshops,” *Computers and Industrial Engineering*, vol. 48, pp. 789-97, 2005.
[21] V. T'Kindt, N. Monmarche, F. Tercinet, and D. Laugt, “An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem,” *European Journal of Operational Research*, vol. 142, pp. 250-7, 2002.
[22] Y. Kuo-Ching and L. Ching Jong, “An ant colony system approach for scheduling problems,” *Production Planning and Control*, vol. 14, pp. 68-75, 2003.
[23] Y. Kuo-Ching and L. Ching-Jong, “An ant colony system for permutation flow-shop sequencing,” *Computers and Operations Research*, vol. 31, pp. 791-801, 2004.
[24] L. Yun-Chia and A. E. Smith, “An ant colony optimization algorithm for the redundancy allocation problem (RAP),” *IEEE Transactions on Reliability*, vol. 53, pp. 417-23, 2004.
[25] G. Di Caro, F. Ducatelle, and L. M. Gambardella, “AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks,” *European Transactions on Telecommunications*, vol. 16, pp. 443-55, 2005.
[26] M. Dorigo, G. Di Caro, and L. M. Gambardella, “Ant algorithms for discrete optimization,” *Artificial Life*, vol. 5, pp. 137-72, 1999.
[27] T. Stutzle and H. H. Hoos, “MAX-MIN Ant System,” *Future Generation Computer Systems*, vol. 16, pp. 889-914, 2000.
[28] M. Dorgio and T. Stutzle, “The ant colony optimization metaheuristic: algorithms, applications, and advances,” in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Norwell, MA: Kluwer Academic Publishers, 2002, pp. 251-285.