Journal of Applied Sciences 7 (2): 208-213, 2007
ISSN 1812-5654
© 2007 Asian Network for Scientific Information

Parallel Machine Scheduling (PMS) in Manufacturing Systems Using the Ant Colonies Optimization Algorithmic Rule

[1]P.V. Senthiil, [1]V. Selladurai and [2]R. Rajesh
[1]Faculty of Mechanical Engineering,
Coimbatore Institute of Technology, Coimbatore-641 014, India
[2]Pittsburgh University, UK

Abstract: This study introduces a new approach for decentralized scheduling in a parallel machine environment based on the ant colonies optimization algorithm. The algorithm extends the use of the traveling salesman problem for scheduling in one single machine, to a multiple machine problem. The results are presented using simple and illustrative examples and show that the algorithm is able to optimize the different scheduling problems. Using the same parameters, the completion time of the tasks is minimized and the processing time of the parallel machines is balanced.

Key words: Distributed artificial intelligence, scheduling algorithms, ant colonies

INTRODUCTION

In the current competitive market environment, companies have to deliver the goods on the date committed to the costumers, using the available resources in the most efficient manner. To achieve these aims, an optimized scheduling is required. The scheduling methods proposed in the beginning of last century by Henry Gantt, have developed into very sophisticated algorithms, focusing both on deterministic and on stochastic systems. Today, analytical solutions for different scheduling problems, as well as heuristic optimization methods like Genetic Algorithms (Tofin, 2003), or market-based-approaches?. However, there are still many topics, both theoretical and practical, which have to be studied in the near future (McKay *et al*., 2001). One of these topics is the distributed scheduling in manufacturing. Scheduling of large-scale processes with complex goals and constrains can hardly be done using a centralized scheduler. The decomposition of the problem distributed by different interacting agents in the process, i.e., resources and tasks, can lead to an optimized solution since every participating part contributes with information and suggestions to the final decision. This type of distributed methodology is even more important since the future scheduling policies will have to include online and reactive rescheduling, in order to face different phenomena like production breakdowns or changes in the production planning, imposed by clients or by the market. In a distributed scheduling problem, the number of agents involved and the quantity of information that has to be exchanged is very large. Multi-agent algorithms based on social insects, can avoid this complexity. Social insects, e.g., ants, have captured the attention of scientists because of the high structuration level that the colonies can achieve, especially when compared to the relative simplicity of the individuals. Artificial ants are presented in detail by Marco *et al*. (1996), as the result of preliminary works, where ants were used to solve different types of NP-hard problems. Robertino and Chenk (2004) and Cicirello and Smith (2001), have proposed the application of the ant colonies algorithm to solve the job shop floor problem. Here, we propose a new approach for decentralized scheduling in a parallel machine environment.

SCHEDULING IN A PARALLEL MACHINE MODEL (PMM)

In many manufacturing and assembly facilities, every job can be processed in the same type of machines. This kind of environment, where the machines are set up in parallel is usually referred to as parallel machines environment (Pinedo and Chao, 2002). The study of this environment, is very important from both a theoretical and a practical point of view: the occurrence of resources in parallel is common in the real world, e.g., in the flexible flow shop configurations and the techniques used for machines in parallel are often used in decomposition procedures for multistage systems.

Corresponding Author: P.V. Senthiil, Faculty of Mechanical Engineering, Coimbatore Institute of Technology, Coimbatore-641 014, India

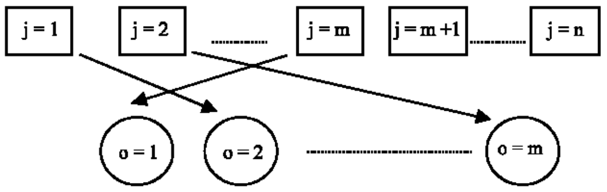

Fig. 1: The parallel machine environment

Modeling the system: Let the number of jobs be denoted by n, where the index j refers to a job and a number of machines in parallel by m, where the index o refers to the machines. Each job j as to be processed at one of the machines o and any machine can do it. Figure 1 shows the general representation of this environment. Each machine has a setup time sij, an interval of time used to take out the last job i, put in the new job j and to change the parameters of the machine if the new job is different from the last one. The processing time poj is the time required by machine o to process the job j. If a list of the n jobs to be executed is known before each production shift, the problem is deterministic. The n jobs can be divided into families and each family divided into types.

Then, each type has to be produced in a certain quantity. The division into families and types is necessary, for example, when the setup times vary from family to family, but the processing times vary from type to type. The objective to be minimized in a scheduling problem is always a function of the completion times of the jobs j in the machines o. This completion time is denoted by Coj or simply Cj when we talk about the completion time of the job j on the last machine. The most common objective functions in this problem are: make span Cmax-it is the completion time of the last job to leave the system. It is defined as the max (C1; : : : ;Cn); lateness Lj - Every job j as a due date rj, the date the job is promised to the client. The lateness of the job is defined as Lj = Cj - rj which is positive when the job j is delivered late and negative when it is completed early. The maximum lateness Lmax is defined as max (L1; : : : ; Ln) and measures the worst violation of due dates. The scheduling in parallel machines can be seen as a two-step process: one has to determine which jobs are allocated to each machine and then, the sequence of the jobs allocated to each machine.

Scheduling optimization using the traveling salesman problem: The system described previously, can be represented by the triplet Pm||sij||Cmax, which tell us in a compact form that the environment is a parallel machine shop with m identical machines (Pm), with setup times in the machines that depend on the job (called sequence dependent setup-times sij) and where the function to minimize is the make span Cmax. This machine

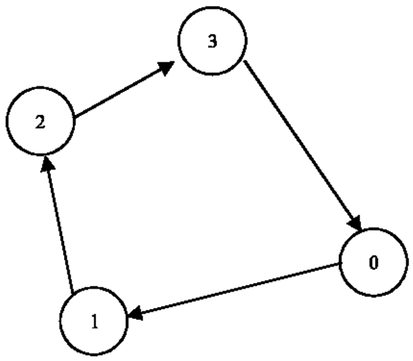

Fig. 2: The TSP in scheduling

Table 1: Setup time from job i to job j

Sij	0	1	2	3
0	0	4	3	5
1	1	0	2	4
2	2	2	0	3
3	2	2	1	0

environment is highly complex. The problem with m = 2 is already a NP-hard problem. Michael (2002) presents a general survey of heuristics to solve this problem, which is of considerable interest to industry. In this study we propose as new heuristic, the extension of the Traveling Salesman Problem (TSP), used to find the optimal scheduling of the make span problem with sequence-dependent setup times for the case of one machine only (Gilmore and Gomory, 1964). In the next sections, we describe the TSP algorithm applied to one machine and then the extension to a multi-machine environment. The TSP is a classical optimization problem, where there are n cities that have to be visited by a salesman. The salesman's objective is to visit all the cities using the shortest or the fastest possible way. To solve the scheduling problem, an analogy between the cities and the n jobs is done, where the distance between the cities are the setup-times between the jobs. One extra city can be included, which is city 0, that corresponds to the initial state of the machine. The path between cities i and j, is in fact the setup-time sij of the machine to change from the job i to the job j. Figure 2 and Table 1 present an example for n = 3. The production time is not relevant in the case of only one machine, even if it is different for each job, since at the end, all jobs will be produced at that machine. In this case, the optimal solution is 0→ 1→ 2→ 3→ 0 with the make span time of

$$C_{max} = 9+\sum_{j=1}^{n} Pj$$

The parallel machine model with TSP. We propose that the parallel machine problem is seen as a multidimensional TSP with m different dimensions. The jobs are once again the cities

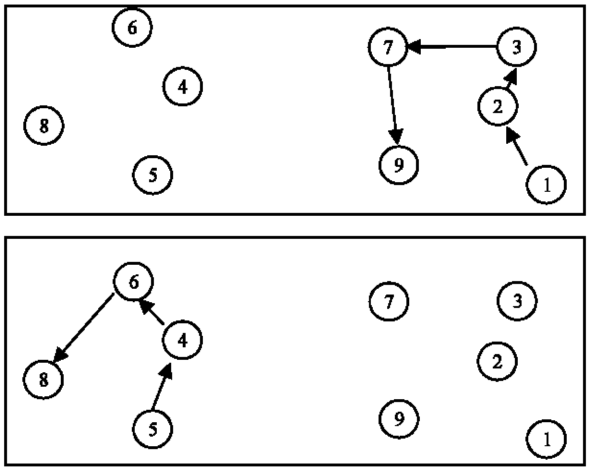

Fig. 3: The multiple TSP

and each machine is a salesman. From the n different jobs, m jobs are randomly allocated to the m machines. Then, for the machine that finishes first a j job, the algorithm chooses the next job to assign from the remaining n-m jobs still to be processed. The second machine to finish a job, has then to choose among the remaining n-m-1 jobs still to execute and so one. At the end of the optimization algorithm, each machine o has its own subset no of jobs of the total n = n1 + n2 + : : : + n_m jobs and has to operate this subset in a way that minimizes the production time. Thus, the TSP not only finds the shortest path between the cities, but it also finds which cities have to be visited by each salesman. One example presenting the algorithm schematically is depicted in Fig. 3. Imagine that there are two machines o = 1 and o = 2 and n = 9 jobs to be processed. The machines can have different processing times. The initial stage is not considered here. Image that at the beginning, jobs j = 1 and j = 5 are assigned to machines o = 1 and o = 2, respectively. The machines start the production and machine o = 1 finishes the job first. Then, it can receive a job from the remaining set [2; 3; 4; 6; 7; 8; 9]. The next job assigned to machine o = 1 is job j = 2. Meanwhile, machine o = 2 ends the first job and can receive a job from the set [3; 4; 6; 7; 8; 9]. Job j = 4 is assigned and while it produces this new job, job j = 3 is assigned to machine o = 1 from the set [3; 6; 7; 8; 9]. Maybe this machine is able to finish job j = 3 before job j = 4 is finished in machine o = 2, so machine o = 1 receives job j = 7 and machine o = 2 receives the job j = 6. Finally, the last jobs [8; 9] are assigned to machines o = 2 and o = 1, respectively. Note that in this case, the production time is relevant for the solution. The TSP problem is one of the most studied optimization problems. There are several different analytical methods and heuristics to solve the problem. However, in this particular case, we have to find not only the best solution to each machine, but also to cluster the orders in the different subsets. This dynamic effect is difficult to integrate in the classical algorithms that solve the TSP. Thus, a distributed optimization algorithm is more suitable to solve this problem than a classical method. Next section introduces the ant colonies distributed optimization algorithm to solve this multiple TSP framework.

SCHEDULING USING SNT COLONIES

Ants are social insects. They live in colonies and all their actions are towards the survival of the colony as a whole, rather than the benefit of a single individual of the society. The individual ants have no special abilities. They communicate between each other using chemical substances, the pheromones. This indirect Communication allows the entire colony to perform complex tasks, such as establishing the shortest route Paths from their nests to feeding sources. In (Marco *et al.*, 1996) an optimization algorithm was proposed that tries to mimic the foraging behavior of real ants, i.e. the behavior of wandering in the search for food. This algorithm has already been successfully used to solve the TSP (Maria and Dorigo, 1996) and other NP hard optimization problems (Silva *et al.*, 2002). The next subsections describe the ant colonies algorithm and a new application in scheduling of production systems.

General description of the ant colonies algorithm: When an ant is searching for the nearest food source and comes across with several possible trails, it tends to choose the trail with the largest concentration of pheromone τ, with a certain probability p. After choosing the trail, it deposits a certain quantity of pheromone, increasing the concentration of pheromones in this trail. The ants return to the nest using always the same path, depositing another portion of pheromone in the way back. Imagine then, that two ants at the same location choose two different trails at the same time. The pheromone concentration on the shortest way will increase faster than the other: the ant that chooses this way, will deposit more pheromones in a smaller period of time, because it returns earlier. If a whole colony of thousands of ants follows this behavior, soon the concentration of pheromone in the shortest path will be much higher than the concentration in other paths. Then the probability of choosing any other way will be very small and only very few ants among the colony will fail to follow the shortest path. There is another phenomenon related with the pheromone concentration. Since it is a chemical substance, it tends to evaporate in the air, so the concentration of pheromones vanishes along the time. In this way, the concentration of the less used paths will be much lower than that on the most used ones, not only because the concentration increases in the other paths, but also because their own concentration decreases. In general, the ant colony behavior can be described formally using the following mathematical framework. Let the nest and the food source be connected by several different paths, connecting n intermediate nodes. The ant k in node i chooses one of the possible trails (i; j) connecting the actual node to one of other possible positions j ϵ [1,.....n], with probability $pij^k = f(\tau_{ij})$ 1

$$ij = f(\tau_{ij})$$

where τ_{ij} is the pheromone concentration on the path connecting i to j, in the way to the food source. The pheromone in this trail will vary in time according to:

$$\tau_{ij}(t+1) = \tau_{ij}(t)^* \rho + \delta_{ij}^{k} \text{........................ } 2$$

where δ_{ik}^{k} is the pheromone released by the ant k on the trail (i; j) and ρ ε [0,1]
is the evaporation coefficient. The system is continuous, so the time acts as the performance index, since the shortest paths will have the pheromone concentration increased in a shorter period of time. This is the mathematical description of a real colony of ants. However, the artificial ants that mimic this behavior can be uploaded with more characteristics, e.g. memory and ability to see. If the pheromone expresses the experience of the colony in the job of finding the shortest path, memory and ability to see, express useful knowledge about the problem the ants are solving. In this way, (1) can be extended to: where ηij is a visibility function and Γ is a tabu list. In this case, the visibility expresses the capability of seeing which is the nearest node j to travel towards the food source. Γ is a list that contains all the trails that the ant has already passed and must not be chosen again (artificial ants can go back before achieving the food source). This acts as the memory of an ant. If the TSP is the problem to be solved, the visibility function can be the inverse of the distance from city i to city j expressed in a matrix dij. Then $\eta ij = 1/dij$ and the tabu list Γ is the list of cities that the ant has already visited. The parameters α and β express the relative weight between the importance of pheromone concentration Γ and the visibility η. Finally, each ant deposits a pheromone δ_{ij}^{k} on the chosen trail: $\delta_{ij}^{k} = \Gamma_c$..... (4) where Γc is a constant. In the artificial ants' framework, Eq. 2 is not sufficient to mimic the increasing pheromone concentration in the shortest path. With real ants, time acts as a performance index, but the artificial ants use all the same time to perform the task, whether they choose a short path or not. For the artificial ants, it is the length l of the paths they have passed that will determine if the solution is good or not. Thus the best solution should increase even more the pheromone concentration on the shortest trail. To do so, (2) is changed to:

$$\Gamma ij(t+n) = \Gamma ij(t)^* \rho + \Delta\tau ij \ (5)$$

where $\Delta\tau ij$ are pheromones deposited in the trails (i; j) followed by all the q
ants, $\Delta\tau ij = \Sigma X\delta ij^{\ k} * f(1/zk)$ .... (6) and zk is the performance index. In the TSP case the zk can be the length $l_k = \Sigma dij$ of the path chosen by the k ant. In this way, the global update is biased by the solution found by each individual ant. The paths followed by the ants that achieved the shortest paths have their pheromone concentration increased.

Notice that the time interval taken by the q ants to do a complete tour is t+n iterations. A tour is a complete route between the nest and the food source and an iteration is a step from i to j done by all the ants. The algorithm runs Nmax times, where in every Nth tour, a new ant colony is released. The total number of iterations is Nmax*n. The general algorithm for the ant colonies is described in Fig. 4.

Ants in the scheduling: To apply the ant colonies in the scheduling of a production system, we use the same mathematical framework described in the previous section. The definition of the matrices for this particular problem is: (1) The matrix dij is not the distance between the towns. When there are no sequence dependent setup-times, matrix dij is given by dij = pj; Vi<n. When there are sequence dependent setup-times, the matrix dij is given by dij = sij + pj; Vi<n. Since there exists m machines, there are m matrices doij, which are the setup times and processing times in each machine o. All the m matrices are equal if the machines are identical. (2) There exist also m matrices $\eta oij = 1$ doij and m matrices τoij. The tabu lists Γk for each ant k will now be a matrix with dimension (n*m), keeping the information about the jobs already executed and the machine o where they were executed. (3) The cost function z is now the make span Cmax. The algorithm is the one presented in Fig. 4, except for the fact that in every iteration n, the ant k has to switch between the m machines, which means, it has to see what will be the next machine o to be available and then choose the next order using the matrices ηoij and τoij.

```
Initialization:
  Set for every paire (i, j): $T_{ij} = T_0$
  Set N = 1 and define a $N_{max}$
  place the q ants

   Build a complete tour
     For i = 1to n
     For k = 1to q
     Choose the nextnode using $p^k_{ij}$ = (3)
     Update locally $T_{ij}$ (t) using (4)
     Update the tabu list T

    Analyze solutions
      For k = 1to q
      Computer performance index z = (lk)
      Update globally $T_{ij}$ (t-n) using (5)
```

Fig. 4: Ant colonies optimization algorithm

SIMULATION RESULTS

The simplest case that can be considered is the P2‖Cmax, which is the problem of two machines in parallel (P2), with no sequence depending setup times, where the cost function is the make span Cmax. This problem is of interest because minimizing the make span has the effect of balancing the load over the various machines, which is an important objective in practice. During the last couple of decades, many heuristics have been developed for this NP-hard problem. One of the heuristics is the Longest Processing Time first (LPT) rule (Graham, 1969), that assigns at t = 0 the largest jobs to the machines. After that, whenever a machine is free, the longest job among those not yet processed is put on the machine. This heuristic tries to place the shortest jobs toward the end of the schedule, where they can be used for balancing the loads. This heuristic is very useful to measure the performance of other heuristics, since it is possible to define a bound relative to an optimal possibly unknown schedule Cmax (OPT), based on Theorem 1 (Michael, 2002).

Theorem 1. For Pm‖Cmax, Cmax (LPT)/Cmax (OPT) < 4/3-1/3 m: The theorem states that is possible to define a minimal bound for the make span Cmax (OPT) of an optimal scheduling, based on the make span Cmax (LPT) defined by the LPT scheduling and on the number of machines m in parallel. Example 1 - No sequence dependent setup-times The first example here analyzed is the case of n = 9 jobs, processed by m = 2 identical machines in parallel, whose processing times are represented in Table 2. The LPT rule here will have a make span of 26, corresponding to the sequences 0 = 1; 1→ 3→ 5→ 7→ 9 and o = 2; 2→ 4→ 6→ 8. Using Theorem 1, it is possible to prove that Cmax(OPT)_ 23. Using the ant algorithm described in section 3, with the parameters q = 9, α = 1, β = 2, ρ = 0: 9, τ0 = 0:001 tuned using a trial and error approach, the solution is given by the following sequences: o = 1; 9→ 8→ 5→ 4→ 6 and o = 2; 1→ 7→ 3→ 2, which have a make span of 24. It is proven that the ant algorithm is able to find an optimized solution, where both machines are equally balanced (Cmax (o = 1) = max (o = 2) = 24). Example 2-Sequence dependent setup-times. The previous example is trivial, since the setup-times are not considered. If the problem being studied is P2‖sij‖Cmax (the same problem of the Example 1, but where there exists sequence depending setup times), then the complexity of the problem increases severely. Let us consider the scenario where there are n = 9 different jobs grouped into 4 different families, which have one common characteristic. The shop has the same 2 machines. The production time for each job is the same has described in Table 2, grouped in four families: family A (j = 1; 2), family B (j = 3; 4), family C (j = 5; 6) and family D (j = 7; 8; 9). The setup-time in each machine depends on if the family changes or not. The setup time is 1 minute if the new job type belongs to the same family and 2 min if it belongs to a different family. The setup time's matrices soij are represented in Table 3. The LPT rule originates a make span of 32, since the resulting sequences are the same as in Example 1, plus the setup-times. Note however, that Theorem 1 does not stand anymore for this application. To apply the ants algorithm, with the same parameters used in the previous example, we have to add the matrices soij to the prior matrices doij. In this case, the ants propose the following sequences: o = 1; 7→ 8→ 9→ 4→ 3 and o = 2; 5→ 6→ 1→ 2. The make span is 29 (Cmax (o = 1) = Cmax (o = 2) = 29) and once again, both machines are balanced.

Table 2: Processing time of job j in machine o

Job j	1	2	3	4	5	6	7	8	9
Poj	7	7	6	6	5	5	4	4	4

Table 3: Setup time from job i to j in machine o

S_{oij}	1	2	3	4	5	6	7	8	9
1	0	1	2	2	2	2	2	2	2
2	1	0	2	2	2	2	2	2	2
3	2	2	0	1	2	2	2	2	2
4	2	2	1	0	2	2	2	2	2
5	2	2	2	2	0	1	2	2	2
6	2	2	2	2	1	0	2	2	2
7	2	2	2	2	2	2	0	1	1
8	2	2	2	2	2	2	1	0	1
9	2	2	2	2	2	2	1	1	0

CONCLUSIONS AND FUTURE WORK

This paper presents a new algorithm to optimize the scheduling in a parallel machine- manufacturing environment. This algorithm has two features: first it extends the optimization of a single machine problem to a parallel machine environment using the TSP and secondly, it uses the ant colonies optimization algorithm to find a solution for this new problem. The ant algorithm explores both the knowledge and the experience of independent agents in the search for the best solution. The ants are able to communicate between each other by means of pheromones, which have embedded the results of prior attempts from other ants, enabling the convergence to an optimal solution. The examples solved in this paper showed that the algorithm is extremely versatile. Changes in the environment are easily introduced in a single matrix. In this way, in a production system managed by this algorithm, changes in the manufacturing processes do not imply a change in the scheduling method. Usually, when the environments conditions change, the scheduling method has to change also, since the schedulers are usually develop for a single problem. As the next step, we will extend the algorithm to the flow shop problem, a serial sequence of parallel machines problem.

REFERENCES

Cicirello, Vincent A. and Stephen F. Smith, 2001. Ant colony for autonomous decentralized shop floor routing. In: Proceedings of ISADS- 2001, Fifth International Symposium on Autonomous Decentralized Systems.

Gilmore, P.C. and R.E. Gomory, 1964. Sequencing a one-state variable machine: A solvable case of the travelling salesman problem. Oper. Manage., 12: 655-679.

Graham, R.L., 1969. Bounds on multiprocessing timing anomalies. SIAM J. Applied Mathematics, 17: 263-269.

Marco, D.V., Maniezzo and A. Colorni, 1996. Ant system: Optimization by a colony of cooperating agents. IEEE Trans. Sys., Man and Cybernetics, Part B 26: 29-41.

Maria, G.L. and M. Dorigo, 1996. Solving symmetric and asymmetric tsps by ant colonies. In: IEEE Conference on Evolutionary Computation (ICEC'96).

McKay, K., M. Pinedo and S. Webster, 2001. A practice-focused agenda for production scheduling research. Production and Operations Management, 14: 231-248.

Pinedo, M., 2002. Scheduling: Theory, Algorithms, Syst. 2nd Edn., Prentice Hall.

Pinedo, P.X. Chao, 2002. Operations Scheduling with Applications in Manufacturing and Services. McGraw-HILL International Ed.,

Robertino, J. and H. Chenk, 2004. Applications of ant colony system for job-shop Scheduling. Jorbel-Belgiuan J. Opera. Res. Stat. Computer Sci., 34: 39-53.

Silva, C.A., T. Runkler, J.M. Sousa and R. Palm, 2002. Optimization of Logistic Processes Using Ant Colonies. In: Proceedings of Agent-Based Simulation Wellman, M.P., W.E. Walsh, P. Wurman and J.K. Mackie-Mason, 2001. Auction Protocols for Decentralized Scheduling. Games and Economic Behavior, 35: 271-303.

Tofin, U., 2003. Sequencing and optimization using random Genetic concepts ORSA J. Computing, 12: 213-221.