An Ant Approach to the Flow Shop Problem

Thomas Stützle
TU Darmstadt, Computer Science Department
Alexanderstr. 10, 64283 Darmstadt
Phone: +49-6151-166651, Fax +49-6151-165326
email: stuetzle@informatik.tu-darmstadt.de

ABSTRACT: In this article we present an Ant Colony Optimization approach to the Flow Shop Problem. ACO is a new algorithmic approach, inspired by the foraging behavior of real ants, that can be applied to the solution of combinatorial optimization problems. Artificial ants are used to construct solutions for Flow Shop Problems that subsequently are improved by a local search procedure. Comparisons with other heuristics for the Flow Shop Scheduling problem show that with our approach very promising results are obtained.

1 Introduction

The Flow Shop Problem (FSP) can be stated as follows: Each of n jobs $1, \ldots, n$ have to be processed on m machines $1, \ldots, m$ in that order. The processing time of job i on machine j is t_{ij}. The processing times are fixed, nonnegative, and may be 0 if a job is not processed on some machine. Further assumptions are that each job can be processed on only one machine at a time, the operations are not preemptable, the jobs are available for processing at time zero and setup times are sequence independent. Here we consider the permutation Flow Shop Problem (FSP), the same job order is chosen on every machine. The objective then is to find a sequence, i.e., a permutation of the numbers $1, \ldots, n$ that minimizes the completion time $C_{\max}$, also called makespan, of the last job. The problem is $\mathcal{NP}$-hard, only some special cases can be solved efficiently (Johnson, 1954). Thus, many approaches have been proposed the find near optimal schedules in reasonable time. Currently available algorithms can be classified as either constructive or improvement methods. Among the constructive heuristics are approaches proposed in (Campbell et al., 1970; Dannenbring, 1977; Nawaz et al., 1983). Of the constructive heuristics the NEH heuristic (Nawaz et al., 1983) seems to be the best performing for a wide variety of problem instances. Improvement approaches are descending local searches (Dannenbring, 1977) and metaheuristics like Simulated Annealing (Osman and Potts, 1989), Tabu Search (Taillard, 1990; Widmer and Hertz, 1989; Nowicki and Smutnicki, 1996; Reeves, 1993), and Genetic Algorithms (Reeves, 1995). A path algorithm approach based on a particular neighborhood structure is presented in (Werner, 1993).

In this article we propose an Any Colony Optimization (ACO) approach to the Flow Shop Problem. Ant Colony Optimization (ACO) is a population based, cooperative search metaphor inspired by the foraging behavior of real ants. One of the underlying ideas of ACO is to use the equivalent of the *pheromone trail* used by real ants as a medium for cooperation and communication among a colony of artificial ants. The seminal work on ACO is Ant System (Dorigo, 1992; Dorigo et al., 1996) that was first proposed for solving the Traveling Salesman Problem (TSP). In Ant System, the ants are simple agents that are used to construct solutions, guided by the pheromone trail and heuristic information based on intercity distances. Since the work on Ant System several extensions of the basic algorithm have been proposed, among those are Ant Colony System (Dorigo and Gambardella, 1997), $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System (Stützle and Hoos, 1998) and the rank-based version of Ant System (Bullnheimer et al., 1997). Here, in particular, we consider the application of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) to the FSP.[1] The performance of ACO approaches with respect to solution quality and convergence speed can be further enhanced by adding a local search phase (Stützle and Hoos, 1997; Dorigo and Gambardella, 1997), in which ants are allowed to improve their solutions by a local search procedure. Thus, we apply $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System to the FSP using a fast local search procedure to improve the solutions constructed by the ants.

The article is structured as follows. We introduce $\mathcal{MM}$AS with its application to the FSP in Section 2 and discuss some

[1] For more work on Ant Colony Optimization we refer the interested reader to the Ant Colony Optimization home page at `http://iridia.ulb.ac.be/dorigo/ACO/ACO.html`.

details on the local search procedure used in Section 3. Then we explain some implementation enhancements and give the parameter settings used for the experimental results that presented in Section 5.

2 $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for the Flow Shop Problem

The FSP is like the TSP[2] a permutation problem in which a permutation of the integers $\{1, \ldots, n\}$ has to be found such that some objective function is minimized. A difference between both problems is that in the TSP only the relative order of the cities is important; a permutation $\pi_1 = (1\ 2\ \ldots\ n)$ represents in some sense the same solution as the permutation $\pi_2 = (n\ 1\ 2\ \ldots\ n-1)$ giving the same objective function value. Yet, in the FSP the absolute position of the jobs is of importance, i.e., the permutations π_1 and π_2 represent really different solutions with, most probably, quite different makespan.

In ACO the pheromone trail of real ants is imitated by some real numbers τ_{ij} associated with *solution attributes*. This pheromone trail is updated during the run of the algorithm. As in the FSP the absolute position of a job is of importance, we use the position of a job as a solution attribute. τ_{ij} then represents the desire of setting job i at the jth position in the sequence. The trails form a kind of adaptive memory of previously found solutions and are modified as described below after every iteration. Actually we should write $\tau_{ij}(t)$, where t is the iteration counter. This should be done to make explicit that the amount of pheromone depends on the current iteration t and changes during the run of the algorithm. Yet, for simplicity we omit the iteration counter. For the construction of a sequence (in this case a permutation of the numbers $\{1, \ldots, n\}$) we introduce a dummy job 0 on which the ants are set initially. The ants construct a sequence by first choosing a job for the first position, then a job for the second position and so on until all jobs are scheduled. To choose a job for a particular position j an ant with a probability p_0 makes the best possible decision, i.e. it chooses for that position a not scheduled job with maximal τ_{ij}. With probability $1 - p_0$ a job for the position j is chosen according to the following probability distribution:[3]

$$p_{ij} = \begin{cases} \frac{\tau_{ij}}{\sum_{i\ not\ scheduled} \tau_{ij}} & \text{if job } i \text{ is not yet scheduled} \\ 0 & otherwise \end{cases} \tag{1}$$

To keep track of the jobs already used in the partial sequence, the ants maintain a list, storing the partial sequence. After a complete sequence is constructed and possibly improved by a local search, the trails are updated. Like in Ant Colony System (Dorigo and Gambardella, 1997), also in $\mathcal{MM}$AS only one ant is allowed to update the trails. The chosen ant lays down a constant quantity $Q = 1$ of pheromone. The trail intensities are updated according to

$$\tau_{ij}^{new} = \rho \cdot \tau_{ij}^{old} + \Delta\tau_{ij} \tag{2}$$

where ρ, with $0 < \rho < 1$, is the persistence of the trail, thus $1 - \rho$ represents the evaporation. The lower ρ, the faster information gathered in previous iterations is forgotten. The amount $\Delta\tau_{ij}$ is equal to $1/C_{\text{best}}$ if job j is placed on position i, otherwise zero, C_{best} being the makespan of the ant that updates the trails. Thus, positions which are often occupied by certain jobs receive an higher amount of pheromone and in the construction phase jobs will be placed preferably on these positions. To give a balance between exploration of new solutions and exploitation of the available information, in $\mathcal{MM}$AS limits on the possible interval of the pheromone trail strength are imposed (Stützle and Hoos, 1998). The interval for the trails is determined by $[\tau_{\min}, \tau_{\max}]$.

3 Local Search for the FSP

Local search for the FSP usually starts from some initial, feasible sequence $\pi \in \Pi$ and then tries to improve it by small changes. The set of possible moves is defined by a neighborhood of the current sequence. Most often, for the FSP the following neighborhood structures are used:

(1) swaps of two neighboring jobs at position i and $i + 1$, $i = 1, \ldots, n - 1$ (*swap-moves*)
(2) exchanges of jobs placed at the ith and the jth position, $i \neq j$ (*interchange-moves*)
(3) remove the job at the ith position and insert it in the jth position (*insertion-moves*)

[2] In the TSP a shortest closed tour through a set of cities with given inter-city distances has to be found.

[3] This rule for the solution construction, making with a certain probability the best possible decision, is called *pseudo-random-proportional rule* and was proposed for Ant Colony System (Dorigo and Gambardella, 1997).

Local search based on swap-moves is very fast, as only a low number of possible moves has to be inspected, yet the obtainable solution quality is rather low and we do not consider it further. In (Taillard, 1990; Osman and Potts, 1989) it was shown that the neighborhood based on insertion-moves can be evaluated more efficiently than the one based on interchange-moves and additionally gives at least the same solution quality. Thus, for the hybrid algorithm combining sequence construction by $\mathcal{MM}$AS and improvements of these sequences by a local search, we use the insertion neighborhood. The moves are defined as follows: Let (i, j) be a pair of positions. The new permutation π^* is obtained by removing the job $\pi(i)$ at position i and inserting it at position j. In case $i < j$ we obtain $\pi^{new} = (\pi(1), \ldots \pi(i-1), \pi(i+1), , \ldots \pi(j), \pi(i), \pi(j+1), \ldots, \pi(n))$ and in case $i > j$ we get $\pi^{new} = (\pi(1), \ldots \pi(j-1), \pi(i), \pi(j), \ldots \pi(i-1), \pi(i+1), \ldots, \pi(n))$. The size of the neighborhood is $(n-1)^2$. Using the fast neighborhood evaluation of (Taillard, 1990), the set of possible moves can be examined in $O(n^2 m)$.

As for larger problem instances the computation times for the local search still grow rather high, we use a modified first-move strategy. For a position i we examine all possibilities for inserting the job $\pi(i)$ and in case an improved schedule is found, we perform the best insertion move found during the neighborhood scan for job $\pi(i)$. During our experiments we could verify that, when starting from random initial solutions, for the same amount of allowed computation time the first-improvement local search gives significantly better results than the best-improvement version. In Figure 1 we outline the algorithmic skeleton of $\mathcal{MM}$AS with local search.

```
procedure $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for FSP
  (1) Initialize the pheromone trails and parameters
  (2) while ( termination condition not met) do
        construct a solution for the ant
        Improve solution by local search
        Update the pheromone trail, $\forall \tau_{ij}\ \tau_{\max} \geq \tau_{ij} \geq \tau_{\min}$
  (3) return best solution found
end $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for FSP;
```

Figure 1: Algorithmic frame for $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System with local search for the FSP.

4 Implementation Details and Parameter Choices

For the sequence construction steps we consider candidate sets if jobs are chosen probabilistically according to Equation 1. The candidate sets of size $cand$ are defined by an analogy to the best sequence π^* found during the run of the algorithm. Within the candidate set are the first $cand$ jobs of π^* that are not yet sequenced. The next job for a specific position is then chosen among the not yet scheduled jobs that are within the neighborhood set according to Equation 1. We found that $cand = 5$ gives a reasonably good performance is obtained.

As we allow only rather short run times for $\mathcal{MM}$AS, we make the search procedure more aggressive by favoring very strongly the exploitation of π^*. To find good solutions fast, we use the sequence as determined by the NEH heuristic(Nawaz et al., 1983) as an initial solution and apply local search to this sequence. This sequence then is chosen for the first update of the trails. Due to this choice our algorithm performs at least as well as the NEH heuristic. In the subsequent iterations we always choose π^*, the best solution found during the run of the algorithm for the trail update. The number of ants is chosen as one, i.e., in each iteration only one ant constructs a solution and applies local search. The number of ants is chosen so low as for most problem instances due to the severe run-time limits only very few iterations can be performed and using more ants would reduce the algorithm to a multi start descent procedure. To have a high exploitation of π^* we choose p_0 very high, fixed as $p_0 = \frac{n-4}{n}$, i.e., with a very high probability the best possible choice is made. This makes $\mathcal{MM}$AS similar to an iterated local search approach (Martin et al., 1992) as the sequences are constructed in such a way that they present slight changes with respect to the best solutions found so far. The other parameters are set throughout all experiments as follows. The trail limits are chosen as $\tau_{\text{max}} = \frac{1}{(1-\rho)} \cdot \frac{1}{\pi^*}$, i.e. τ_{max} corresponds to its algebraically maximal value, and $\tau_{\text{min}} = \tau_{\text{max}}/5$. Initially the trails are set to their maximally possible value τ_{max} and the persistence of the trail is chosen as $\rho = 0.75$.

5 Experimental Results

For an experimental evaluation of our ACO approach to the FSP we compare the average solution quality obtained on problem instances taken from (Taillard, 1993) with $\mathcal{MM}$AS to the NEH heuristic (Nawaz et al., 1983), NEH followed by a subsequent phase of local search (NEH+ls) using the same local search procedure as for $\mathcal{MM}$AS, our implementation of the Simulated Annealing approach proposed in (Osman and Potts, 1989) (SAOP), and to multiple descent starting from randomly generated solutions (MD). For $\mathcal{MM}$AS and MD we allowed the same computation time as needed by Simulated Annealing. Thus, only few local searches especially for problems involving many machines can be performed with $\mathcal{MM}$AS and MD. The computational results are presented in Table 1. Note that the computation times for NEH and NEH+ls are significantly lower than those needed by SAOP and $\mathcal{MM}$AS.

As can be seen $\mathcal{MM}$AS performs best for all problem sizes. A surprising result is that even the MD approach outperforms the Simulated Annealing algorithm, contradicting computational results presented, e.g., in (Osman and Potts, 1989). We could verify that this is due to the first-improvement pivoting rule in our local search algorithm and the fast implementation of the local search due to the improvement suggested by (Taillard, 1990). Note, that on average the first-improvement and the best-improvement version of our local search based on insertion-moves yield roughly the same solution quality. Using a first-improvement approach allows to apply much more often a local search and so better results are obtainable. If using a best-improvement pivoting rule the results for MD are significantly worse those of SAOP for most problem instances. Thus, an important part of the success of $\mathcal{MM}$AS is due to the fast first-improvement local search procedure. The initialization of trails, as discussed in Section 4, using the solution given by the NEH heuristic with subsequent local search seems to be especially helpful for the larger instances with $n \geq 100$. We verified that for the smaller FSP instances with $n \leq 50$ this kind of initialization does not help to improve solution quality.

Table 1: Results on $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for short runs on FSP instances from (Taillard, 1993). The run times for Simulated Annealing (SAOP) (Osman and Potts, 1989), $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) and multiple descent (MD) refer to seconds on a Sun Sparc 5 Workstation and are indicated by `time`. The Nawaz-Enscore-Ham heuristic (NEH) (Nawaz et al., 1983) and NEH with local descent require shorter computation times. The computational results are present as average percentage excess over the best known solutions. The results of the best performing algorithm for each problem size is indicated in boldface.

`instances`	NEH	NEH + ls	SAOP	$\mathcal{MM}$AS	MD	`time`
ta001 - ta010 (20×5)	3.663	1.923	1.061	**0.408**	0.730	0.33
ta011 - ta020 (20×10)	4.601	2.453	1.462	**0.591**	0.955	1.33
ta021 - ta030 (20×20)	3.731	2.417	1.116	**0.410**	0.628	3.30
ta031 - ta040 (50×5)	0.727	0.258	0.597	**0.145**	0.255	0.90
ta041 - ta050 (50×10)	5.073	3.438	3.012	**2.193**	2.873	3.30
ta051 - ta060 (50×20)	5.971	4.267	3.147	**2.475**	3.138	9.00
ta061 - ta070 (100×5)	0.527	0.324	0.509	**0.196**	0.219	1.90
ta071 - ta080 (100×10)	2.215	1.250	1.719	**0.928**	1.226	7.00
ta081 - ta090 (100×20)	4.275	2.942	3.249	**2.238**	3.229	20.50

The best performing heuristic algorithm for the FSP is the sophisticated Tabu Search algorithm of Nowicki and Smutnicki (Nowicki and Smutnicki, 1996). Yet, this algorithm is fine-tuned to the FSP exploiting special features of the FSP to speed up the local search. On the contrary, our approach is a minor variation of earlier applications of $\mathcal{MM}$AS to the TSP. The main difference to this previous application to the TSP is the solution construction and, of course, the local search algorithm.

6 Conclusion

Applications of Ant Colony Optimization to Scheduling problems are rather rare. A first application of Ant System to the Job-Shop Scheduling Problem is proposed in (Colorni et al., 1994), yet no competitive results with state-of-the-art algorithms are obtained. In this paper we introduced an ACO algorithm, based on $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, to the Flow Shop Scheduling Problem. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System is enhanced by a fast local search procedure to yield high quality solutions to the FSP. We showed that our approach gives high quality solutions to FSPs in short time performing better or at least comparable to other state-of-the-art algorithms proposed for the FSP. These first results of the application of an Ant Colony Optimization approach to the Flow Shop Problem are very promising and suggest that ACO algorithms may be applied successfully to scheduling problems.

References

Bullnheimer, B., Hartl, R. F., and Strauss, C. (1997). A New Rank Based Version of the Ant System — A Computational Study. Technical report, University of Viena, Institute of Management Science.

Campbell, H., Dudek, R., and Smith, M. (1970). A Heuristic Algorithm for the n Job, m Machine Sequencing Problem. *Management Science*, 16(10):B–630–B–637.

Colorni, A., Dorigo, M., Maniezzo, V., and Trubian, M. (1994). Ant System for Job-Shop Scheduling. *Belgian Journal of Operations Research, Statistics and Combuter Science*, 34(1):39–53.

Dannenbring, D. (1977). An Evaluation of Flow Shop Sequencing Heuristics. *Management Science*, 23(11):1174–1182.

Dorigo, M. (1992). *Optimization, Learning, and Natural Algorithms*. PhD thesis, Politecnico di Milano.

Dorigo, M. and Gambardella, L. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41.

Johnson, S. (1954). Optimal Two- and Three-stage Production Scheduling with Setup Times Included. *Naval Reseach Logistics Quarterly*, 1:61–68.

Martin, O., Otto, S., and Felten, E. (1992). Large-step Markov Chains for the TSP Incorporating Local Search Heuristics. *Operations Research Letters*, 11:219–224.

Nawaz, M., Enscore Jr, E., and Ham, I. (1983). A Heuristic Algorithm for the m-Machine, n-Job flow-Shop Sequencing Problem. *OMEGA*, 11(1):91–95.

Nowicki, E. and Smutnicki, C. (1996). A Fast Tabu Search Algorithm for the Permutation Flow-Shop Problem. *European Journal of Operational Research*, 91:160–175.

Osman, I. and Potts, C. (1989). Simulated Annealing for Permutation Flow-Shop Scheduling. *OMEGA*, 17(6):551–557.

Reeves, C. (1993). Improving the Efficiency of Tabu Search for Machine Sequencing Problems. *Journal of the Operational Research Society*, 44(4):375–382.

Reeves, C. (1995). A Genetic Algorithm for Flowshop Sequencing. *Computers & Operations Research*, 22(1):5–13.

Stützle, T. and Hoos, H. (1997). The $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and Local Search for the Traveling Salesman Problem. In *Proceedings of ICEC'97*, pages 309–314.

Stützle, T. and Hoos, H. (1998). Improvements on the Ant System: Introducing $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. In G.D. Smith, N.C. Steele, R. A., editor, *Artificial Neural Networks and Genetic Algorithms*, pages 245–249.

Taillard, E. (1990). Some Efficient Heuristic Methods for the Flow Shop Sequencing Problem. *European Journal of Operational Research*, 47:65–74.

Taillard, E. (1993). Benchmarks for Basic Scheduling Problems. *European Journal of Operational Research*, 64:278–285.

Werner, F. (1993). On the Heuristic Solution of the Permutation Flow Shop Problem by Path Algorithms. *Computers & Operations Research*, 20(7):707–722.

Widmer, M. and Hertz, A. (1989). A New Heuristic Method for the Flow Shop Sequencing Problem. *European Journal of Operational Research*, 41:186–193.