An Ant Colony Optimization for Solving the HFFS problem

A. Sioud, C. Gagné and M. Gravel

Département d'informatique et de mathématique, Université du Québec à Chicoutimi
{asioud, mgravel, c3gagne}@uqac.ca

1 Introduction

In many industries including pharmaceutical, metallurgical production, electronics, ceramics and automotive manufacturing, there are frequently setup times on equipment between two different activities. These setup times can be or not sequence dependent. Dudek et al. [3] reported that 70% of industrial activities include dependent setup times. Production of good schedules often relies on management of these setup times [1].

Furthermore, the real production systems rarely employ a single machine. Therefore, in many times, the regular flowshop problem is extended with a set of usually identical parallel machines at each stage, i.e., instead of having a series of machines, we have a series of stages. The goal here is to increase the capacity and the outflow of the production system and to reduce the impact of bottleneck stages on the overall shop efficiency. It is also frequent in practice to have optional treatments for products, like polishing or additional decorations in ceramic manufacturing as an example [5]. In this latter case some jobs will skip some stages. Thereby, we obtain the hybrid flexible flowshop noted as HFFS. This present paper considers the sequence dependent setup times hybrid flexible flowshop problem (SDST/HFFS) with the objective of minimizing the makespan and noted as $((PM)^{(i)})_{i=1}^{m}/F_j, s_{ijk}/C_{max}$.

In this work, to solve the SDST/HFFS problem, we introduce a new ant colony optimization algorithm (ACO) which incorporates a transition rule that having feature using look-ahead information based on heuristic and past information based on archive concept such as the multiobjective evolutionary computation.

2 ACO for the HFFS problem

Only a few papers addressed SDST/HFFS problem where different approaches have been proposed to solve the studied problem such as simulated annealing (SA),integer programming (IP), random keys genetic algorithm (RKGA), immune algorithm (IA) and iterated local search (ILS) [4] which represents the best approach.

In this paper we adapt the ACO of Dorigo and Gambardella [2] including a modified transition rule called the pseudo-random-proportional rule, global and local trail updating rules, use of restricted candidates list and the use of local improvement rule. The defined transition rule uses past (pheromone trail), present (visibility) and future (look ahead) informations. Indeed, the past information is introduced by a matrix built from an archive that stores the best solutions throughout the evolution process as in some cases in multi-objective evolutionary algorithms using the Pareto-optimal concept. The visibility is represented by the relative setup times between jobs. Whereas the look ahead information use an heuristic that anticipates the choices in the transition rule. This heuristic is based on an upper bound of the makespan. This heuristic will be also used in the local improvement step.

3 Computational results and discussion

The benchmark problem set consists of 960 problem tests available from http://soa.iti.es. The instances are combinations of N and M, where $N = \{20, 50, 80, 150\}$ and $M = \{2, 4, 8\}$. The processing times are generated from a uniform [1, 99] distribution. The setup times are generated according to four distributions [1, 25], [1, 50], [1, 99] and [1, 125]. This corresponds to a ratio between setup and processing times of 25%, 50%, 100% and 125%, respectively. Concerning the number of parallel machines at each stage, there is a group with two parallel machines per stage and

groups where the number of parallel machines at each stage is sampled from a uniform distribution in the range [1, 4]. The probability of skipping a stage for each job is set at 0.10 and 0.40. All the experiments were run 20 times on an Intel Core 2.4 GHz processors and 4 GB of main memory with a stopping criterion set to $n^2 \times m \times 1.5\ ms$ elapsed CPU time [4].

Table 1 compares the results of different approaches. All the presented results show the average deviation to the best known solution grouped by $n \times m$ and where the best average results are bold. The ILS and GAR columns show the results of the iterated local search of Naderi et al. [4] and the genetic algorithm of ruiz and Maroto [5], respectively. The ACO and the ACO_I column present the ACO algorithm with and without the transition, respectively.

Table 1. Comparison of different approaches

Instance	ILS	GAR	ACO	ACO_I
20×2	**1.39**	2.61	2.05	1.57
20×4	**1.27**	2.94	1.95	1.31
20×8	**1.49**	3.50	1.56	1.53
50×2	**1.26**	2.51	2.45	1,45
50×4	**1.85**	4.03	1.97	1.88
50×8	1.75	3.20	1.78	**1.71**
80×2	2.06	2.15	2.14	**2.02**
80×4	2.92	4.64	3.05	**2.88**
80×8	5.46	4.51	4.65	**4.33**
120×2	3.38	3.42	3,35	**3.32**
120×4	6.74	5.97	5,94	**5,88**
120×8	9.25	5.19	6,21	**5.02**
Average	3.23	3.72	3,10	**2.95**

The first observation is that the ACO algorithm outperform the GAR algorithm on all the instances except for the 80×8 and 120×8 group instances where the deviation is very small. Also, the ACO algorithm perform better than the ILS on the larger instances. Our hypothesis for these results is that the two elements explore more search space when they have more information. Indeed, the two elements, *i.e.*, the look ahead and the past information, allow to take into account the job positions on the next stages and the job position on the best found sequences.

As can be seen, the ACO_I algorithm which embeds the new local improvement heuristic provides better results among all the other algorithms except for the 20 jobs instance group where the ILS performs little better. Furthermore, the ACO_I algorithm obtain the best average for eight instance groups. Comparing to the ILS algorithm, the ACO_I found results very close to the ILS results on the smaller instance groups. Moreover, the ACO_I algorithm have the best average (2.95). So, we can conclude that, in general, the local improvement heuristic is effective.

4 Conclusion

In this work, we have introduced an ACO algorithm that integrates archive concept in the transition rule and look-ahead information to solve the hybrid flexible flowshop problem with sequence-dependent setup times minimizing the makespan. The proposed approach is essentially based on adapting the transition rule to the specifics of the studied problem. The numerical experiments allowed us to demonstrate the efficiency of the proposed approach for this problem, especially for large instance groups.

A perspective of this work is to use this proposed approach for other scheduling problems in particular and other optimization problems in general, specially real-world problems. Also, we will work on the refinement of the look-ahead heuristic to enhance results quality.

References

1. Allahverdi, A., Ng, C.T., Cheng, T.C.E., Kovalyov,M.Y.: A survey of scheduling problems with setup times or costs. European Journal of Operational Research. 187(3), 985 – 1032 (2008)

2. Dorigo M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation (1997)
3. Dudek, R.A., Smith, M.L., Panwalkar, S.S.: Use of a case study in sequencing/scheduling research. Omega 2(2), 253–261 (1974)
4. Naderi, R., Ruiz, R., Zandieh, M.: Algorithms for a realistic variant of flowshop scheduling. Comput. Oper. Res. 37(2),236–246 (February 2010)
5. R. Ruiz and C. Maroto. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, 169(3):781–800, March 2006.