Bio-inspired algorithm for the Two-Machine Scheduling Problem with a Single Server

Jean-Paul Arnaout
Gulf University for Science & Technology
Mishref, Kuwait
+(965) 2530 7314
arnaout.j@gust.edu.kw

ABSTRACT
Within the arena of Swarm Intelligence, this research introduces a bio-inspired ant colony optimization (ACO) algorithm for solving the NP-hard Two-Machine Scheduling Problem with a Single Server. The problem consist of a given set of jobs to be scheduled on two identical parallel machines, where each job must be processed on one of the machines, and prior to processing, the job is set up on its machine using one server; the latter is shared between the two machines. ACO performance was compared to the exact solution (B&B), as well as Genetic Algorithm, and the computational results reflected the superiority of *ACO* in all tested problems. Furthermore, this superiority improved as problem sizes increased, while solving the tested problems within a reasonable computational time.

Categories and Subject Descriptors
I.2.8 [**Problem Solving, Control Methods, and Search**]: Subject descriptor: *Heuristic methods*.

General Terms
Algorithms, Management, Experimentation.

Keywords
Ant Colony Optimization, B&B, Parallel Machines

1. INTRODUCTION
The problem addressed in this paper is the scheduling of N available jobs on two parallel machines with a single server to minimize the makespan, C_{max}, without preemption. In particular, two operations are needed to schedule the N jobs on the two machines, where a single server that can handle one job at a time will carry out the first operation; i.e. the setup. The second operation (processing) is carried out on the machine immediately following setup but without the server. This problem has been addressed in the research literature and industry including manufacturing and service industry. We refer to this problem hereafter as $P2,S1||C_{max}$. Considering that the problem $P2,S1|s_j = s|C_{max}$ is strongly NP-hard [6], then $P2,S1||C_{max}$ is also strongly NP-hard.

Due to the complexity of the problem, finding optimal solutions for large problems is very time consuming, and therefore, heuristic algorithms become the most practical way of solving the problem.

Abdekhodaee [1] addressed the $P2,S1|s_j|C_{max}$ problem, where the authors introduced a mixed integer programming formulation. The authors extended their study in [2] to include a greedy heuristic, a genetic algorithm and the Gilmore-Gomory algorithm also for the general case. Gan [5] extended on [2] by considering the problem under study, $P2,S1||C_{max}$, where they developed a branch-and-price algorithm, The authors also presented an integer programming formulation for the same problem. Hasani [7] suggested several mixed integer programming formulations, based on blocks and setups, which turned out to be superior to the algorithms presented in [2] and [5]. Hasani [8] developed a simulated annealing and a genetic algorithm and the obtained results showed superiority to all previous algorithms and models proposed in the recent literature.

In this paper, a newly designed Ant Colony Optimization (ACO) algorithm is introduced for the problem and the results are compared to those obtained by the genetic algorithm in [8] as well as an exact algorithm (B&B).

The rest of this paper is organized as follows. In Section 2, we discuss the IP formulation of the problem. In Sections 3 and 4, the Ant Colony Optimization algorithm is presented in general then introduced for this problem in particular. The computational tests are presented in Section 5 and finally we conclude this research in Section 6.

2. INTEGER PROGRAMMING FORMULATION
The IP formulation below was adopted from [5]. It will be solved using a Branch and Bound as explained in Section 5.

We will not go into details about the formulation here, and the reader can refer to [5] for a comprehensive description.

BICT 2014, December 01-03, Boston, United States

DOI 10.4108/icst.bict.2014.257914

$$\min C_{max}$$

subject to

$$C_{max} \geq T_k + S_k + P_k, \quad k = 1, \ldots, n$$

$$\sum_{i=1}^{n} x_{ki} = 1, \quad k = 1, \ldots, n$$

$$\sum_{k=1}^{n} x_{ki} = 1, \quad i = 1, \ldots, n$$

$$S_k = \sum_{i=1}^{n} x_{ki} s_i, \quad P_k = \sum_{i=1}^{n} x_{ki} p_i, \quad k = 1, \ldots, n$$

$$T_k \geq T_{k-1} + S_{k-1}, \quad k = 2, \ldots, n$$

$$T_j \geq T_k + S_k + P_k - M z_{jk}, \quad j,k = 1, \ldots, n;\ k < j,\ M \text{ large}$$

$$z_{jk}^{+} - z_{jk}^{-} = y_j - y_k, \quad z_{jk} = z_{jk}^{+} + z_{jk}^{-}, \quad j,k = 1, \ldots, n$$

$$\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{z}^{+}, \mathbf{z}^{-} \in \{0, 1\}, \quad \mathbf{T}, \mathbf{S} \geq 0$$

3. ANT COLONY OPTIMIZATION (ACO)

Following the natural ability of ants to find the shortest path between their nest and food places, scholars were inspired to simulate this behavior to solve combinatorial optimization problems. ACO was firstly proposed by [3], with the first algorithm aiming to search for an optimal path in a graph. Since then, many studies have emerged on the implementation of ACO to solve a wider range of numerical problems. For more on ACO literature, the reader can refer to [4].

An example that illustrates the ants' ability to find the shortest path is as follows. If there were two closed paths that a group of ants can take, where one would be longer than the other, initially (at time $t = 0$), almost 50% of the ants will take the short path and the rest will take the long path. During their moves, they deposit a chemical substance (pheromone). Ant instinct to follow each other by sensing the pheromone attracts them to move to the path that has more pheromone. Since pheromone evaporates quicker in the long path than the short one, the pheromone amount in the shorter path will be more condensed, which will attract more ants to that path over time.

To solve an optimization problem using ACO, the problem can be represented as a *connected graph* (nodes and edges). Initially, a certain amount of pheromone is deposited in each edge in the graph. After the ants move from a node to another according to the *State Transition Rule*; which determines the probability of an ant to move from one node to another. The probability is determined by two factors: pheromone amount (τ) and visibility (η). The probability of moving from node i to j for ant k can be calculated as follows: $P_{ij}^{\ k} = \frac{\tau_{ij}^{\ \alpha} \eta_{ij}^{\ \beta}}{\sum_{l \in \Psi} \tau_{il}^{\ \alpha} \eta_{il}^{\ \beta}}$, where Ψ represents the set of non-visited nodes and (η) is usually determined by a *greedy rule* (heuristic). Two important parameters to direct the search are α and β; which are the exponents in the probability function that determine the importance of the pheromone amount over the visibility amount. After all ants construct their tours, pheromone will be updated locally and globally according to the quality of the constructed tour solutions (objective function value) and evaporation rate (ρ) as shown in (1) and (2). Please note that the ACO variant used here is AS (Ant System). The first term accounts for the pheromone reduction (local update) due to evaporation, and the second term (global update) represents the additional deposit of pheromone in the links (arcs) according to the produced tour length when traveled by all ants.

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{\ k} \qquad \forall (i,j) \tag{1}$$

$$\Delta\tau_{ij}^{\ k} = \begin{cases} 1/l^k & \text{if arc } (i,j) \text{ is used by ant } k \\ 0 & \text{Otherwise} \end{cases} \tag{2}$$

where: l^k represents the length of the tour taken by ant k.

4. ACO APPLICATION TO *P2,S1||C*$_{max}$

Before we present the ACO for the problem under study, it is important to note that the solution is obtained when the optimal sequence is generated. In other words, we do not need to report the sequence on each machine separately, because the first job in the sequence will be scheduled on machine 1, following which the jobs according to their sequence will be processed on the machine where they can start first. To illustrate this, consider a set of 5 jobs with the following sequence: {J3, J1, J5, J4, J2}, then their assignment to the machines is shown in Figure 1, where the shaded blocks refer to setup time.

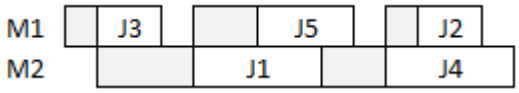

Figure 1. Example of a sequence

The solution is represented by a vector (S) that contains N entries representing the number of jobs and each entry is populated with a job number (j = 1,…,N) in order to represent the sequence. The Pheromone trail (τ_{ij}) is defined for this stage to indicate the favorability of sequencing job j after job i, where the indices (0, j) refer to the job scheduled first and (j,N) refer to the job scheduled last. In addition to the pheromone, we assist the algorithm with (η_{ij}) to solve the problem.

Initially, each ant needs to decide on the job in the first position; i.e. position $S[0]$, by means of $\eta_{0,j} = 1/s_j$, in order to give more importance to the job with the smaller setup time. Next, the ant decides on the job in the last position ($S[N\text{-}1]$), where in this case $\eta_{j,N} = 1/p_j$, so that the job with the smallest processing gets the priority.

After the assignment of the first and last position, the rest of the sequence is populated with the remaining jobs using an adaptation of the Min-Idle algorithm in [9] as follows.

Define St_M, where M = 1,2, referring to the earliest possible start of the next job on the associated machine. St_M is calculated as the

max between the completion of the last assigned job on a machine and end of the setup on the second machine. For example, in Figure 2, St_1 is 7 because no job can start on machine 1 before the setup finishes on machine 2, and $St_2 = 12$ as this is the completion time of the last job assigned. Next, L_M will represent the overlap on every machine; in particular, $L_1 = St_2 - St_1$ and $L_2 = St_1 - St_2$. Subsequently, $L_{max} = \max\{L_1, L_2\}$ and the associated machine with this maximum will be the one assigned next. In Figure 2, $L_1 = 5$, $L_2 = -5$, and $L_{max} = 5$, indicating that the next job will be scheduled on machine 1.

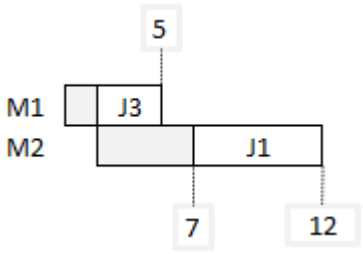

Figure 2. Highlighting St_M and L_{max}

Having said this, the ant will decide on the job *j* to be sequenced after the last scheduled job *i* based on Equation 3.

$$\eta_{ij} = {}^{1}\!\Big/\!\max\{(L_{max} - s_j), 0\} \tag{3}$$

The rationale behind Equation 3 is to try to fit the setup of the next job within the existing overlap. If a job *j*'s setup does not fit, then the corresponding $\eta_{ij} = 0$. In the case none of the jobs' setup can fit, then $\eta_{ij} = {}^{1}\!/\!_{s_j}$ is used in order to give more priority to the job with the smallest setup.

The probability to sequence job *i* after *j* for ant *k* is calculated using Equation 4.

$$\Pi_{ij}^{k} = \frac{(\tau_{ij})^{\alpha}.(\eta_{ij})^{\beta}}{\sum_{l \epsilon \Psi}(\tau_{ij})^{\alpha}.(\eta_{ij})^{\beta}} \tag{4}$$

After all ants finish their paths, we update the pheromone amounts in each link locally by reducing the amounts due to evaporation. We also update the pheromone amounts in each link globally by increasing the ones constructed by the ant that produced the least C_{max}. This is estimated according to Equation 5, where the lower bound (LB) = $0.5 * \left(\sum_{j \epsilon N}(s_j + p_j) + \min_{j \epsilon N}\{s_j\}\right)$ as suggested by [2].

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \phi.\Delta\tau^{Best}{}_{ij} \tag{5}$$

where:

$$\Delta\tau^{Best}{}_{ij} = \begin{cases} LB / C\max^{Best} & \text{if arc } (i, j) \text{ is used by best ant} \\ 0 & \text{Otherwise} \end{cases}$$

5. COMPUTATIONAL TESTS

The proposed ACO was implemented in Microsoft Visual C++ 2010 running on Windows XP with a Pentium 4 processor. The ACO parameters' values used for the preliminary tests conducted in this paper are as follows: $\tau_{ij} = 1,\ \alpha = \beta = 1,\ \rho = 0.01,$ $\phi = 0.2,$ and $No\_of\_Ants = 30.$

ACO was compared to Genetic Algorithms (GA) and Branch and Bound (B&B). The GA results were obtained from [8]. The MIP presented in Section 2 was solved using B&B; the latter's solver used was Lingo 11.0 from Lindo Systems.

For a fair comparison with GA, the data was obtained from [8] where it was generated using p_j = U(0,100) and s_j = U(0,50). Additionally, all algorithms were compared based on the average values of the relation ${}^{Cmax}\!/\!_{LB}$. Furthermore, the same time limits that were used in [8] were adopted.

The results are shown in Table 1 and they clearly indicate that ACO performed the best, followed by GA, with B&B performing the worst. In particular, as the problem size increased, both ACO and GA were able to get the optimal solution, while B&B was not even able to solve the 250 jobs instances. As ACO and GA's performance was close, Figure 3 depicts the percent deviation of GA from ACO, which shows that ACO did perform better.

Table 1. Computational Results for all Algorithms

		Jobs (*N*)			
		8	**20**	**100**	**250**
Algorithms	**B&B**	1	1.02503	1.91226	N/A
	GA	1.00033	1.0004	1	1.00006
	ACO	1	1.0004	1	1
	Time	300s	750s	3600s	3600s

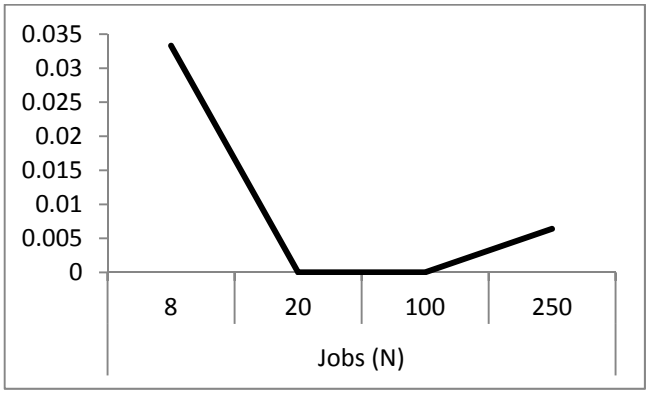

Figure 3. GA's percent deviation from ACO

6. CONCLUSIONS AND FUTURE RESEARCH

In this paper, we have introduced an ant colony optimization algorithm (ACO) for the Two-Machine Scheduling Problem with a Single Server and an objective of minimizing the makespan. The algorithm was compared to Genetic Algorithms (GA) and Branch

and Bound (B&B). The computational tests that were carried out between the rules proved the superiority of the ACO, with the GA performing second, and B&B third.

While the conducted preliminary tests show very promising results, it is worth noting that the rules were compared using one set of uniform distribution for the processing p_j and setup s_j times. An extension to this work would be to test ACO under different distributions, especially with dominant setup times. Moreover, a Design of Experiments will be conducted to determine the best parameters for the introduced ACO which will enable the attainment of superior results. Also, an addition of a local search to ACO is imperative in order to reach better solutions. Finally, it would be interesting to compare ACO to more advanced algorithms than the ones in this paper.

7. REFERENCES

[1] Abdekhodaee, A. & Wirth, A., 2002. Scheduling parallel machines with a single server: Some solvable cases and heuristics. *Computers & Operations Research* 29, 295-315.

[2] Abdekhodaee, A., Wirth, A. & Gan, H.S., 2006. Scheduling two parallel machines with a single server: The general case. *Computers & Operations Research* 33, 994-1009.

[3] Dorigo, M. 1992. Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italie.

[4] Dorigo, M. and Stützle, T. 2004. *Ant Colony Optimization*. MIT Press.

[5] Gan, H.S., Wirth, A. and Abdekhodaee, A., 2012. A branch-and-price algorithm for the general case of scheduling parallel machines with a single server. *Computers & Operations Research* 39, 2242-2247.

[6] Hall, N.G., Potts, C.N. and Sriskandarajah, C., 2000. Parallel machine scheduling with a common server *Discrete Applied Mathematics,* 120, 223-243.

[7] Hasani, K., Kravchenko, S.A. & Werner, F., 2014a. Block models for scheduling jobs on two parallel machines with a single server. *Computers & Operations Research,* 41, 94-97.

[8] Hasani, K., Kravchenko, S.A. & Werner, F., 2014b. Simulated annealing and genetic algorithms for the two-machine scheduling problem with a single server. *International Journal of Production Research* (DOI: 10.1080/00207543.2013.874607).

[9] Hasani, K., Kravchenko, S.A. & Werner, F.. Minimizing the makespan for the Two-Machine Scheduling Problem with a Single Server: Two algorithms for Very Large Instances. Draft 1.0.