Annals of Operations Research (2020) 285:273–293
https://doi.org/10.1007/s10479-019-03138-w

S.I. : PROJECT MANAGEMENT AND SCHEDULING 2018

A worm optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times

Jean-Paul Arnaout[1]

Published online: 26 February 2019

Abstract
This paper addresses the unrelated parallel machine scheduling problem with setup times, with an objective of minimizing the makespan. The machines are unrelated in the sense that the processing speed depends on the job being executed and not the machine. Each job will have different processing times for each of the available machines, is available at the beginning of the scheduling horizon, and can be processed on any of the machines but needs to be processed by one machine only. Sequence-dependent and machine-dependent setup times are also considered. A Worm Optimization (WO) algorithm is introduced and is applied to this NP-hard problem. The novel WO is based on the behaviors of the worm, which is a nematode with only 302 neurons. Nevertheless, these neurons allow worms to achieve several intricate behaviors including finding food, interchanging between solitary and social foraging styles, alternating between dwelling and roaming, and entering a type of stasis/declining stage. WO's performance is evaluated by comparing its solutions to solutions of six other known metaheuristics for the problem under study, and the extensive computational results indicated that the proposed WO performs best.

Keywords Worm optimization · Unrelated parallel machines · Setup time

1 Introduction

This paper addresses the scheduling of N available jobs on M unrelated machines (R_M) using a novel Worm Optimization algorithm (WO) that emulates the worms' behaviors and an objective of minimizing the makespan, C_{max}, without preemption.

With its negligible neurons (only 3.02e−7% of human brain neurons), the worm remarkably is able to realize critical survival activities, especially in relation to switching between social and solitary food searching, locating nutrients, evading unhealthy food, varying between "roaming—global search" and "dwelling—local search", and converting from a reproductive stage to a declining one. WO was first introduced in Arnaout (2016) for the

✉ Jean-Paul Arnaout
arnaout.j@gust.edu.kw

1 Gulf University for Science and Technology, Mishref, Kuwait

traveling salesman problem, where it was compared to ant colony system (ACS), particle swarm optimization (PSO), and genetic algorithm (GA). The computational tests indicated that WO outperformed the algorithms in all problems, as well as attained the optimal solution in all cases. In a later study, Arnaout (2017) developed a WO for the Multiple Level Warehouse Layout Problem and compared it to GA and ACO. The results showed that WO performed better than the other algorithms, especially in large problems.

In the Unrelated Parallel Machine Scheduling Problem (PMSP), the jobs' processing times (P_{ik}: Processing time for job i on machine k) depend on the machine to which they are assigned, and there is no relationship between machine speeds. We consider in this paper sequence-dependent setup times S_{ijk} where the time necessary to set up for two consecutive jobs i and j on machine k may be different if i and j are reversed (i.e. $S_{ijk} \neq S_{jik}$). We also address a more generic form of the problem by assuming that the setup times are machine-dependent (i.e., each machine has its own matrix of $N \times N$ setup times). We refer to this problem hereafter as $R_M|S_{ijk}|C_{max}$. The identical parallel machine scheduling problem $P_M\|C_{max}$, which is simpler special case of $R_M|S_{ijk}|C_{max}$, is an NP-hard even when $M = 2$ (Karp 1972; Garey and Johnson 1979). Subsequently, the latter is also NP-hard; i.e. it is computationally impractical to solve using exact approaches, and heuristic algorithms are more appropriate.

The related literature describes several algorithms for the PMSP but without considering the setup time, where fewer works were conducted. For extensive literature about the problem without setup time, the reader can refer to Arnaout et al. (2010, 2014). Having said that, Al-Salem (2004) introduced the Partitioning Heuristic (PH) to solve large instances of $R_M|S_{ijk}|C_{max}$ and so did Helal et al. (2006) who developed a tabu search TS for the same problem and demonstrated that it performed better than the PH. In Rabadi et al. (2006), the authors solved the problem using a heuristic called Meta-Heuristic for Randomized Priority Search (Meta-RaPS) and showed that their heuristic outperformed the PH. Arnaout et al. (2010) introduced a two-stage Ant Colony Optimization (ACO) for $R_M|S_{ijk}|C_{max}$ and showed its superiority over PH (Al-Salem 2004), TS (Helal et al. 2006) and Meta-RaPS (Rabadi et al. 2006). Ying et al. (2012) developed for the same problem Simulated Annealing (SA) and restrictive simulated annealing (RSA) algorithms. The latter employs a restricted search strategy to eliminate ineffective job moves for finding the best neighbourhood solution. The authors compared their algorithms to the ACO in Arnaout et al. (2010) using the same data sets, and their results indicated the superiority of their algorithms, with RSA performing significantly better than SA. Chang and Chen (2011) developed a set of dominance properties with Genetic Algorithms, introduced a new metaheuristic and reported efficient solutions. Eroglu et al. (2014) proposed a genetic algorithm (GA) for the same problem, and their solutions show that it outperformed ACO (Arnaout et al. 2010) in most combinations as well as the metaheuristic developed by Chang and Chen (2011). Finally, Lin and Ying (2014) presented a hybrid artificial bee colony (HABC) for the problem and showed that it outperformed the best-so-far algorithms such as ACO, TS, RSA, and Meta-RaPS.

Recent works also dealt with the same problem from an exact, heuristic and hybrid perspectives. Wang et al. (2016) presented a hybrid estimation of distribution algorithm (EDA) with iterated greedy (IG) search (EDA-IG), and compared it to former GAs. The authors also proposed a probability model that is based on the neighbor relations of the jobs, in order to assist the proposed algorithm to generate new solutions by sampling a promising search region. An immune-inspired algorithm was proposed by Diana et al. (2015). The authors generated the initial population using the Greedy Randomized Adaptive Search Procedure (GRASP), used Variable Neighborhood Descent (VND) as local search heuristic, and proposed a population re-selection operator. The authors noted that their algorithm performed better than some of the existing ones in the literature. Avalos-Rosales et al. (2015) proposed a new makespan linearization and several mixed integer formulations for this problem, and noted that these models are able to solve larger instances and in a faster computational time. The authors also proposed a metaheuristic algorithm based on multi-start algorithm and VND. The algorithm's performance was improved using composite movements for the improvement phase. Ezugwu et al. (2018) developed an improved symbiotic organisms search (SOS) algorithm with a new solution representation and decoding procedure to make it suitable for the combinatorial aspect of the problem at hand. The authors adapted the longest processing time first (LPT) rule to design a machine assignment heuristic that assigns processing machines to jobs based on the machine dynamic load-balancing mechanism. The heuristic scheme was incorporated into SOS, which led to improved results.

Due to space and scope limitations, we will not address in this paper the large body of knowledge related to the problem at hand with different objectives and constraints. In the latter case, the reader can refer to Allahverdi (2015) for a detailed review.

We develop in this paper a WO algorithm to find high quality solutions for $R_M|S_{ijk}|C_{max}$. Its performance is evaluated by comparing its solutions to the ones generated by TS in Helal et al. (2006), ACO in Arnaout et al. (2010), RSA in Ying et al. (2012), GA in Eroglu et al. (2014), and ABC/HABC in Lin and Ying (2014). All instances and solutions for the addressed problem are available at *SchedulingResearch.com*.

2 Worm optimization

As introduced in Arnaout (2016), WO simulates the worm's behaviors by mimicking its ability to find food, avoid toxins, search in groups or independently, fluctuate between local and global exploration, and convert from a reproductive stage to a declining one. In order to solve an optimization problem using WO, the problem must be represented as a graph (nodes and arcs), where the worm will move from one node to another in order to create a solution.

WO starts by depositing pheromone (τ_{ij}) on all the arcs in the graph. Initially, the worms are social, where the neuron *RMG* responsible of the foraging behavior (Social "1" or Solitary "0") is initialized to 1. Under Social behavior, the worms move between nodes based on a greedy rule and an attraction to the pheromone. This behavior is similar to the one exhibited by ants in ant colony optimization (Dorigo and Gambardella 1997), coupled with a unique attribute for worms, which is toxins avoidance. In particular, a social worm t will move between nodes i and j following the probability in (1):

$$P_{ij}^{t} = \frac{\tau_{ij}\eta_{ij}^{\beta} ADF_{ij}}{\sum_{h\in\Psi} \tau_{ih}\eta_{ih}^{\beta} ADF_{ih}}, \tag{1}$$

where η_{ij} refers to the greedy rule,ADF_{ij} is the bad solution factor for toxins avoidance, Ψ is the set of unvisited nodes, and β is the exponent that determines the importance of the pheromone amount over the greedy rule.

Every iteration consists of a group of worms, *Worms*, completing their path through the network, with the cost of every worm path calculated. In addition, every worm upon completion of its path, has a probability of conducting local search (dwelling). At the end of the iteration, the arcs belonging to the best worm are updated by increasing their pheromone using Eq. (2), and the arcs of the worst worm are updated by decreasing their pheromone (Eq. (3)) as well as potentially adding the worm's path to the bad solutions' list (*ADF*).

$$\tau_{ij} \leftarrow \tau_{ij} \times (1+\rho) \; if \; arc\,(i,j)\, is \; used \; by \;\; BestWorm \tag{2}$$

$$\tau_{ij} \leftarrow \tau_{ij} \times (1-\rho) \; if\; arc\,(i,j)\, is\; used\; by\; WorstWorm \tag{3}$$

$$\text{where } \rho = \frac{WorstWorm - BestWorm}{BestWorm}$$

If the solution does not improve after a predefined number of iterations (*BestIter*), the worms will shift to a solitary foraging behavior by setting *RMG* to 0. Under solitary search, worms behave as the opposite of a social one; i.e. they randomly move between nodes and are repelled by pheromone.

Finally, the number of worms as well as the solution improvement are analyzed at the end of every iteration, following which WO interchanges between a propagative phase (where more worms are produced) and a declining phase, referred to as Dauer (where the number of worms is decreased).

2.1 Solving the $R_M|S_{ijk}|C_{max}$ using WO

$R_M|S_{ijk}|C_{max}$ is modeled using two stages, with a separate pheromone trail and *ADF* for each. Jobs are assigned to machines in the first stage, and their sequence on each machine is determined in the second stage. As highlighted earlier, WO starts with a social foraging behavior ($RMG = 1$), and will alternate to a solitary one ($RMG = 0$) after *BestIter* iterations without improvement. This oscillation between social and solitary is repeated until WO terminates.

In the first stage, WO assigns job j to the kth machine, according to the pheromone trail τ^I_{jk}, the visibility amount η^I_{jk}, and the bad solution factor ADF^I_{jk}. The visibility of this stage is shown in Eq. (4), in which for social worms, η^I_{jk} favours the allocation of a machine that takes the least processing time of job j where P_{jk} refers to the processing time of job j on machine k; i.e. social worms follow a greedy rule. On the other hand, for solitary worms, an equal amount is given to all machines, to ensure a random dispersion of worms.

$$\eta^I_{jk} = \begin{cases} {}^1\!/_{P_{jk}}, & if\; social\; worm; \\ {}^1\!/_{M}, & if\; solitary\; worm \end{cases} \tag{4}$$

Next, worm t assigns job j to machine k according to the probability in Eq. (5). As can be seen, solitary worms are repelled by pheromone $\left({}^1\!/_{\tau^I_{jk}}\right)$ while social ones are lured to it.

$$\Pi^{t,I}_{jk} = \begin{cases} \dfrac{\left(\tau^I_{jk}\right).\left(\eta^I_{jk}\right)^{\beta}.\left(ADF^I_{jk}\right)}{\sum_{h\Psi}\left(\tau^I_{jh}\right).\left(\eta^I_{jh}\right)^{\beta}.\left(ADF^I_{jh}\right)}, & if\; social\; worm; \\ \dfrac{\left({}^1\!/_{\tau^I_{jk}}\right).\left(\eta^I_{jk}\right)^{\beta}.\left(ADF^I_{jk}\right)}{\sum_{h\Psi}\left({}^1\!/_{\tau^I_{jh}}\right).\left(\eta^I_{jh}\right)^{\beta}.\left(ADF^I_{jh}\right)}, & if\; solitary\; worm \end{cases} \tag{5}$$

Following the assignment of jobs to machines in Stage 1, the jobs' sequence is determined in Stage 2. In particular, the probability for job j to be processed after job i on machine k is given in Eq. (6) and the greedy rule is calculated using Eq. (7), where J_k refers to the number of jobs assigned to machine k. In the case of social worms, the greedy rule's rationale is to give more priority to the job that takes the least amount of setup time after processing job i on machine k. Again, a random dispersion is given to the solitary worms.

$$\Pi_{ij}^{k^{t},II} = \begin{cases} \dfrac{\left(\tau_{ij}^{k^{II}}\right)\cdot\left(\eta_{ij}^{k^{II}}\right)^{\beta}\cdot\left(ADF_{ij}^{k^{II}}\right)}{\sum_{h\Psi}\left(\tau_{ih}^{I}\right)\cdot\left(\eta_{ih}^{k^{II}}\right)^{\beta}\cdot\left(ADF_{ih}^{k^{II}}\right)}, & if\ social\ worm; \\ \dfrac{\left(1\big/\tau_{ij}^{k^{II}}\right)\cdot\left(\eta_{ij}^{k^{II}}\right)^{\beta}\cdot\left(ADF_{ij}^{k^{II}}\right)}{\sum_{h\Psi}\left(1\big/\tau_{ih}^{k^{II}}\right)\cdot\left(\eta_{ih}^{k^{II}}\right)^{\beta}\cdot\left(ADF_{ih}^{k^{II}}\right)}, & if\ solitary\ worm \end{cases} \tag{6}$$

$$\eta_{ij}^{k^{II}} = \begin{cases} 1\big/s_{ijk}, & if\ social\ worm; \\ 1\big/J_k, & if\ solitary\ worm \end{cases} \tag{7}$$

After each worm finishes its path, the latter cost (C_{max}) is calculated. Tables 1 and 2 provide sample outputs of Stage 1 and Stage 2, respectively, for a problem with 10 jobs (N = 10) and 4 machines ($M = 4$). In particular, Table 1 shows a one-dimensional array of size N cells. Each cell is populated by the job-machine assignment where the index of the array represents the machine number to which a job in the array cell is assigned; i.e. jobs 1, 5, and 10 are assigned to machine 1, jobs 3 and 6 to machine 2, etc. Table 2 shows a two-dimensional array of size ($M \times N$), where extending on Table 1, job 5 is sequenced first on machine 1, followed by job 1 then job 10, job 3 is sequenced first on machine 2, followed by job 6, and so on.

2.1.1 WO local search: dwelling

For every worm in an iteration, and once it completes its path and generates a C_{max} value, the probability to conduct a local search is determined according to amphid interneuron (AIY), which refers to the ratio of local search. In particular, a random variable is generated, and if its value is less than or equal to AIY, the worm will conduct local search. A simple swapping rule is used for the local search, where two jobs from the worm path are randomly chosen and their positions are interchanged. Extending on the example in Tables 2 and 3 shows three different solutions obtained using local search.

Table 1 Stage 1 output

1	4	2	3	1	2	4	3	3	1

Table 2 Stage 2 output

5	1	10	0	0	0	0	0	0	0
3	6	0	0	0	0	0	0	0	0
4	8	9	0	0	0	0	0	0	0
7	2	0	0	0	0	0	0	0	0

Table 3 Local search examples

5	1	**3**	**9**	1	10	5	1	10
10	6	0	3	6	0	3	**7**	0
4	8	9	4	8	**5**	4	8	9
7	2	0	7	2	0	**6**	2	0

Bold refers to interchanged jobs

2.1.2 WO bad solution list (ADF)

Once all *Worm* in a particular iteration complete their path, WO captures the solutions of the *BestWorm* and *WorstWorm*. The latter's solution is added to the *ADF* list if it meets the criteria highlighted in this section.

The following must be defined for the *Bad Solution List*: ($ADFList_{\sqrt{Worm}, M\times N+1}$), which stores the inferior solutions. The list is dynamic as it stores $\sqrt{Worm}$ solutions, where $Worm$ is changing from one iteration to another depending on the dauer status (reproductive versus declining population). The reason behind $\sqrt{Worm}$ is not to burden WO's computational load. ($M \times N$) is needed to store in each row (machine) a job sequence of size N. The last entry ($M \times N + 1$) is needed to store the *WormCost* (C_{max}). $ADFList$ is also linked to two arrays, ADF^{I}_{jk} and $ADF^{k^{II}}_{ij}$ (one for each stage). Once a solution is added to the list, its associated arcs are updated following $ADF^{I}_{jk} = ADF^{k^{II}}_{ij} = \lambda$, where λ is the bad solution factor that ensures that the probability of these arcs to be reassigned is reduced. As an example, assuming that the solution presented in Table 2 was the one for *WorstWorm* in an iteration with $C_{max} = 8000$, then it will be added as a row to $ADFList$ as shown in Table 4.

The pseudo code for the *ADF* modeling is shown below:

Step 1: Sort *ADFList* in the descending order according to *WormCost*
Step 2: Assess the iteration's *WorstWorm*'s solution

Step 2.1: If (*WorstWorm* > $ADFList_{1,M\times N+1}$):
- add worm's tour to *ADFList*,
- update the worm's associated arcs pheromones:

$$\tau^{I}_{jk} = \tau^{I}_{jk} * (1-\rho),\ if\ arc\ (j,k)\ is\ used\ by\ WorstWorm$$

$$\tau^{k^{II}}_{ij} = \tau^{k^{II}}_{ij} * (1-\rho),\ if\ arc\ (i,j)\ is\ used\ by\ WorstWorm$$

Step 2.2: ElseIf (*WorstWorm* > $ADFList_{\sqrt{Worm}, M\times N+1}$), replace row $\sqrt{Worm}$ in the list with the current worm's tour, and update the pheromone as in Step 2.1.

Step 3: for ($w = 1, \ldots, \sqrt{Worm}$; $k = 1, \ldots, M$; $j = 1, \ldots N$),

Step 3.1: If ($ADFList_{w,[((K-1)\times N)+j]}$ <>0), then: $ADF^{I}_{lk} = ADF^{k^{II}}_{lh} = \lambda$; where $l = ADFList_{w,[((K-1)\times N)+j]}$; $h = ADFList_{w,[((K-1)\times N)+j-1]}$.

Following Step 1, $ADFList_{1,M\times N+1}$ and $ADFList_{\sqrt{Worm}, M\times N+1}$ will store the worst solution so far and the least worst solution, respectively. In Step 2, *ADFList* is updated according to *WorstWorm* quality. Note that in Step 2.1, if the solution generated (i.e. C_{max}) was worse than the worst solution, the pheromone of the worm's tour is reduced. In Step 2.2, and in case the solution generated was worse than the last tour stored in *ADFList* (*which represents the least inferior solution*), then the latter is released from *ADFList* back into the search space and replaced with this worm's solution. In Step 3, *ADF* values are updated based on the solutions that are present in *ADFList*.

Table 4 ADFList representation

5	1	10	0	...	0	3	6	0	...	0	4	8	9	0	...	0	7	2	0	...	0	8000
Sequence of jobs on Machine 1						*Sequence of jobs on Machine 2*					*Sequence of jobs on Machine 3*						*Sequence of jobs on Machine 4*					*WormCost*

2.1.3 WO dauer stage modeling

As mentioned earlier, worms transition between a reproductive and a declining/Dauer phase. In the reproductive stage, more worms are produced in every iteration, while in the Dauer phase, worms decline after every iteration. Eventually the algorithm terminates when *Worm* $= 0$.

At the beginning of every iteration, the foraging behavior of worms is determined based on the solutions/food quality (*FQ*). WO starts with a social foraging style and $FQ = 0$ which refers to good solution quality. After a pre-determined number of iterations (*BestIter*) is exceeded without any solution improvement, the solution quality is marked as bad ($FQ = 1$) and foraging is switched to the alternate behavior (social or solitary). *FQ* is reset to zero when a forthcoming better solution is realized.

The number of tours in the algorithm is tracked by tallying the worms in every iteration, with a maximum level set at *MaxIteration*, indicating a high concentration of worms. Equation (8) is used to normalize the concentration to a (0,1) scale, where $WCt = 1$ indicates that WO reached the maximum number of iterations.

$$Worms\,Concentration(WCt) = \frac{Tours}{Max\,Iteration} \tag{8}$$

Following each iteration, the Dauer status (*DauerStatus*) is assessed as follows:

Step 1: Compute *FQ* and *WCt*.
Step 2: Compute Dauer level: $DauerStatus = \frac{FQ+WCt}{2}$.
Step 3: Assess if worms are in reproductive or declining stage:

Step 3.1: If ($DauerStatus < 1$), then $Worm = \min\{MaxWorm, Worm * (1 + \phi)\}$,
Step 3.2: If ($DauerStatus == 1$), then $Worm = Worm * (1 - \phi)$.
where ϕ is the rate of reproduction/decrease of worms.

Step 4: If ($Worm == 0$), Stop *WO* and report *BestCost*.

2.1.4 WO summary

The *WO* pseudo code to solve the $\boldsymbol{R_M|S_{ijk}|C_{max}}$ can be summarized as follows:

Step 1: Initialize WO Parameters and populate τ^{I}_{jk}, $\tau^{k^{II}}_{ij}$, ADF^{I}_{jk}, and $ADF^{k^{II}}_{ij}$ with the specified amounts.
Step 2: While ($Worm \neq 0$), Do:

Step 2.1: *Iteration* = *Iteration* + 1;
Step 2.1.1: Decide if worms will follow a social or solitary behavior, based on *RMG*
Step 2.1.2: For a total of *Worm* tours, Do:
Step 2.1.2.1: Assign jobs to machines according to Sect. 2.1 (*WO Stage 1*)

Step 2.1.2.2: Sequence the jobs on each machine according to Sect. 2.1 (*WO Stage 2*)
Step 2.1.2.3: Execute the local search approach according to Sect. 2.1.1 (via *AIY*)
Step 2.1.2.4: Find *WormCost* associated with every *Worm*
Step 2.1.3: Generate from the *Worm* tours the *WorstWorm* and *BestWorm*
Step 2.1.4: Update the pheromone according to Eqs. (2) and (3)
Step 2.1.5: Execute the bad solutions approach described in Sect. 2.1.2 and update the pheromone
Step 2.1.6: Update *Worm* number according to the Dauer approach in Sect. 2.1.3

Step 3: Output the best tour and its cost, Stop WO.

Figure 1 summarizes the steps of WO to solve the $\boldsymbol{R_M|S_{ijk}|C_{max}}$.

2.2 WO originality

As highlighted above, the introduced algorithm, WO, is instigated by elements of the foraging behaviors of worms. The behaviors inspired the design of a search procedure that simulates worms searching the feasible region of the problem at hand for the optimal solution. A few characteristics of WO might resemble present metaheuristics in the swarm intelligence domain, especially ACO, as ants and social worms share an analogous foraging behavior. However, as highlighted in Brabazon and McGarraghy (2018), WO features the below worms' unique behaviors:

- a dual social and solitary foraging style, which ensures that more search space is visited;
- recognizing food quality with a preference to higher-quality sources;
- alternating between dwelling (local search) or roaming (global search) depending on the quality of food;
- ability to avoid toxins (bad solutions formerly visited); and
- engaging in dormancy (Dauer) if environmental conditions are poor (i.e. low quality solutions and high concentration of pheromone) or, alternatively, reproduce if conditions are good.

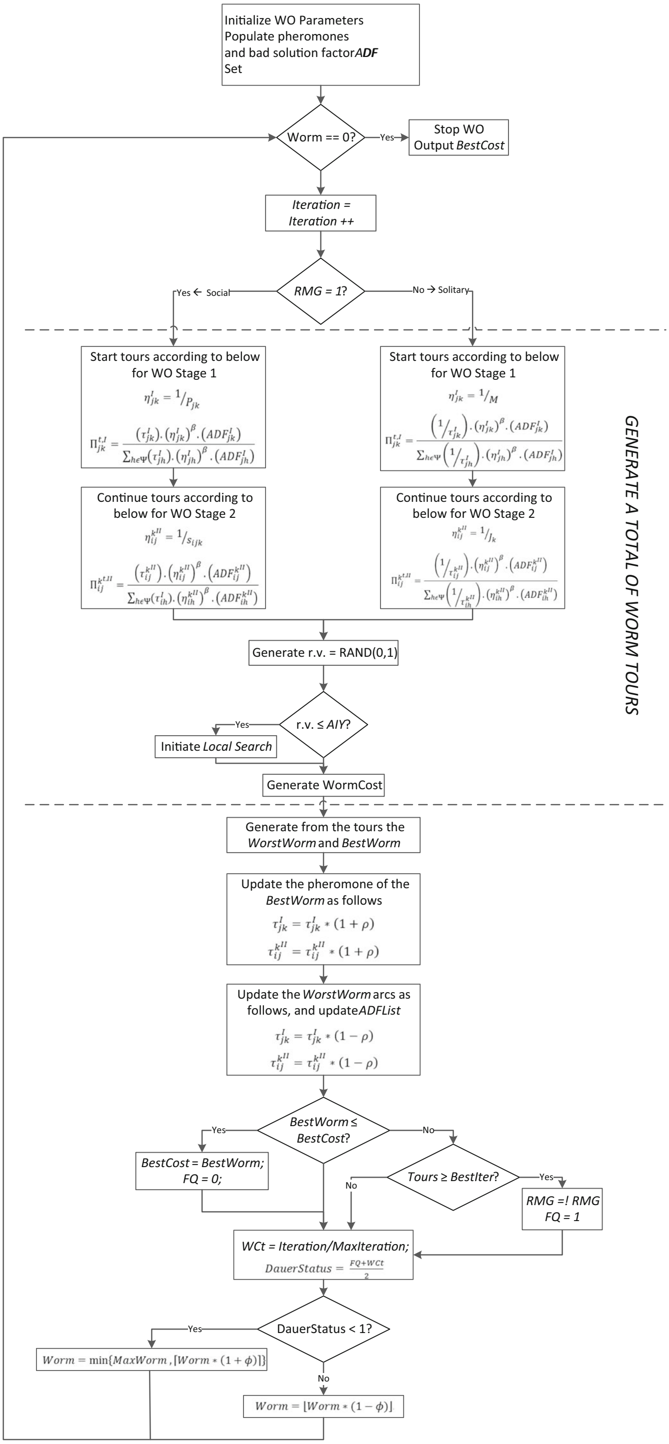

Fig. 1 Flowchart the $R_M|S_{ijk}|C_{max}$

3 Computational tests

The proposed WO was implemented in Microsoft Visual Studio 12.0, with 4 GB of memory available for working storage on a personal computer Intel (R) Core (TM) i3-370 M CPU

Table 5 WO parameters

	Parameter	Range/value	Description
Fixed parameters *Not included in DoE*	$Worm$	10	Worm number: number of worms in the algorithm (at the initialization stage)
	RMG	{0, 1}	Lever between social and solitary strains: $RMG = 1$ indicates only social behavior and $RMG = 0$ only solitary
	τ_{ij}	0.01	Pheromone: initial amount of pheromone deposited on arc (i,j)
	λ	0.01	Bad solution factor: initially, arc attractiveness (ADF) for all arcs equals 1; i.e. each arc has an equal probability of being selected. In the case of a bad solution, it's associated arcs will be assigned an ADF = λ to decrease its selection probability
	ρ	N/A	Pheromone update factor, calculated during algorithm's run as $\rho = \frac{WorstWorm - BestWorm}{BestWorm}$, where $WorstWorm$ and $BestWorm$ refer to the best and worst worms (*in terms of costs*) in a tour
DoE factors	$MaxWorm$	(20, 500)	During the reproductive stage, the number of worms cannot exceed $MaxWorm$
	$MaxIteration$	(5000, 15,000)	Number of tours: referring the total number of worms' tours generated in *WO*
	AIY	(0.1, 0.5)	Percentage of local search: the higher AIY, the higher the chance of dwelling (local search)
	$AIYIter$	(5, 60)	Local search iterations: indicating the max number of local search iterations
	$BestIter$	(100, 1000)	Number of solutions without improvement before concluding that food quality is bad
	ϕ	(0.01, 0.4)	Production rate: rate of reproduction of worms when they are not in Dauer Stage
	β	$\left(1/3, 3\right)$	Exponent to determine the importance of the greedy rule over the pheromone: if $\beta = 1/3$, pheromone is three times more important; if $\beta = 3$, greedy rule is three times more important

Table 6 Summary of Fit of the model

R^2	0.994432
R^2 adj	0.945711
Root mean square error	53.09639
Mean of response	597.6000
Observations (or sum wgts)	40

Table 7 Analysis of Variance of the model

Source	DF	Sum of squares	Mean square	F ratio
Model	35	2,013,996.7	57,542.8	20.4108
Error	4	11,276.9	2819.2	**Prob > F**
C. total	39	2,025,273.6		0.0047

@ 2.4 GHz. Design of Experiments (DoE) is used to decide on the best values for the WO parameters that will minimize the makespan *Cmax*. Numerous publications provide a good review of DoE (e.g., Fisher 1960; Taguchi 1993; NIST/SEMATECH 2006).

The factors considered in this experiment along with their description and levels of low, medium and high are shown in Table 5. In addition, WO parameters with constant values (not included in DOE) are also highlighted in the Table. The values of the parameter levels were selected based on many runs under different settings. Subsequently, JMP 11.0 from SAS was used to generate a D-Optimal custom design, with 40 experiments. The factors along with their interactions were analysed using regression, ANOVA, and factors' effect tests. Three-factor interactions and higher were not considered as they typically have weak effect (Ross 1996).

The *Summary of Fit* of the model and the *Analysis of Variance* are shown in Tables 6 and 7, respectively. The results indicate a good model fit based on the high R^2 and small *p* value for the overall model.

Based on a 95% Confidence Interval, a relatively large t-Stat, and a small p value (less than 0.05), a prediction expression was generated and solved for the minimum *Cmax* while varying the factors' values. As a result, the following parameter values were determined to provide the best performance for WO: $MaxWorm = 450$, $AIY = 0.1$, $AIYIter = 60$, $\beta = 1.74$, $\phi = 0.01$, $BestIter = 432$, and $MaxIteration = 14{,}000$.

WO was compared to Ant Colony Optimization (ACO), Tabu Search (TS), Restrictive Simulated Annealing (RSA), Artificial Bee Colony and its hybrid version (ABS, HABS), and Genetic Algorithm with Local Search (GALA). The ACO results were obtained from Arnaout et al. (2014), TS results from Helal et al. (2006), RSA from Ying et al. (2012), GALA from Eroglu et al. (2014), ABS and HABS from Lin and Ying (2014). All algorithms used the same benchmarking data from Rabadi et al. (2006), where the processing and setup times were randomly generated from two uniform distributions: U(50, 100) and U(125, 175). The data is also available via the scheduling research website (http://schedulingresearch.com/).

Table 8 Average deviations from LB for all test instances

m	n	TS			ACO			RSA			ABC			HABC			GALA	WO		
		B	P	S	B	P	S	B	P	S	B	P	S	B	P	S	B	B	P	S
2	20	6.66	3.97	4.15	4.38	2.59	2.57	4.45	2.61	2.51	4.22	2.53	2.41	**4.14**	**2.47**	**2.37**	4.25	**4.14**	**2.47**	**2.37**
	40	6.05	3.55	3.95	2.27	1.53	1.67	2.84	1.68	1.71	2.59	1.60	1.70	2.37	1.45	1.58	3.46	**2.23**	**1.40**	**1.53**
	60	6.45	3.92	3.77	1.84	**1.08**	1.19	2.53	1.61	1.61	2.33	1.44	1.50	2.10	1.30	1.33	3.74	**1.82**	**1.08**	**1.18**
	80	5.95	3.72	3.8	**1.41**	**0.9**	**0.87**	2.39	1.41	1.24	2.13	1.46	1.30	1.94	1.20	1.17	3.63	**1.41**	**0.90**	**0.87**
	100	6.21	4.13	3.76	**1.35**	2.72	1.57	2.03	1.31	1.28	2.01	1.32	1.26	1.80	1.19	1.09	3.45	**1.35**	**1.16**	**1.07**
	120	6.27	3.84	3.98	**1.06**	1.56	1.47	1.95	1.24	1.17	1.82	1.13	1.16	1.69	1.05	1.06	3.21	**1.06**	**1.01**	**1.04**
4	20	11.1	6.31	6.25	10.1	5.89	6.04	8.74	5.24	5.15	8.67	5.17	5.01	8.48	**5.09**	**4.99**	8.73	**8.47**	**5.09**	**4.99**
	40	8.97	5.14	5.43	7.08	3.95	4.23	5.74	3.35	3.52	5.42	3.23	3.35	5.13	3.00	3.11	7.34	**5.07**	**2.98**	**3.10**
	60	8.17	5.06	5.34	5.28	2.98	3.39	4.75	2.73	2.89	4.51	2.70	2.90	4.08	2.42	2.60	6.44	**4.05**	**2.38**	**2.58**
	80	7.66	4.73	5.1	4.41	2.67	2.7	4.59	2.48	2.41	4.22	2.45	2.52	3.88	2.30	**2.22**	6.20	**3.83**	**2.25**	**2.22**
	100	7.06	4.76	5.37	3.98	2.33	2.42	4.08	2.35	2.34	4.01	2.30	2.35	3.45	2.00	2.05	5.54	**3.40**	**1.98**	**2.04**
	120	6.8	4.71	4.52	3.46	2.39	2	3.62	2.51	2.19	3.71	2.20	2.12	3.33	2.01	1.90	5.29	**3.10**	**1.98**	**1.82**
6	20	24	22.4	22.5	24.9	22.3	22.6	23.12	21.66	**21.70**	23.08	**21.61**	**21.70**	**23.05**	**21.61**	**21.70**	23.07	**23.05**	**21.61**	**21.70**
	40	12.2	9.74	9.69	12.4	9.06	8.99	9.37	7.35	7.30	9.27	7.20	7.31	8.77	7.02	7.04	10.37	**8.61**	**6.79**	**6.95**
	60	10.1	6.38	6	8.59	5.1	4.99	6.51	3.74	3.82	6.18	3.66	3.68	5.79	**3.23**	3.42	8.49	**5.74**	**3.23**	**3.40**
	80	9.76	7.98	7.54	8.13	7.03	6.71	6.91	5.53	5.40	6.57	5.47	5.25	5.95	5.34	5.12	8.03	**5.89**	**5.24**	**5.01**
	100	8.84	6.32	6.15	6.42	4.4	4.61	5.63	3.78	3.72	5.51	3.61	3.65	**4.84**	3.40	3.33	7.13	4.86	**3.35**	**3.31**
	120	8.21	5.76	5.43	5.44	3.3	3.61	5.37	3.13	2.90	4.86	2.86	2.93	4.57	2.53	**2.52**	6.95	**4.52**	**2.52**	**2.52**
8	20	28.3	25.4	25.1	30.1	25.8	26.2	27.14	**24.39**	24.26	27.19	24.42	24.19	**27.04**	**24.39**	**24.08**	27.29	**27.04**	**24.39**	**24.08**
	40	11.1	6.79	6.92	13.1	7.56	7.87	9.03	5.57	5.65	9.21	5.36	5.21	8.10	**4.82**	4.90	11.24	**8.07**	**4.82**	**4.89**
	60	12.8	10.5	10.5	12.9	10.7	10.7	10.49	9.13	8.59	10.38	8.66	8.56	9.62	8.57	8.50	11.48	**9.60**	**8.49**	**8.24**
	80	10.6	6.33	5.84	8.48	5.71	5.27	6.90	4.10	4.00	6.62	3.94	3.87	**6.00**	**3.50**	3.50	9.22	**6.00**	**3.50**	**3.49**

Table 8 continued

m	*n*	TS			ACO			RSA			ABC			HABC			GALA	WO		
		B	*P*	*S*	*B*	*P*	*S*	*B*	*P*	*S*	*B*	*P*	*S*	*B*	*P*	*S*	*B*	*B*	*P*	*S*
	100	10.2	7.76	7.9	9.4	7.09	7.12	7.53	5.84	5.65	7.34	5.70	5.73	6.72	5.51	5.35	8.78	**6.67**	**5.40**	**5.32**
	120	10.1	6.41	6.38	6.7	4.01	4.08	6.66	3.35	3.35	6.04	3.29	3.41	5.33	**2.89**	**2.96**	8.41	**5.24**	2.91	**2.96**
10	20	23.4	15.4	14.2	19.8	12.3	11.9	15.79	9.59	9.22	16.64	9.68	9.38	**15.13**	**9.16**	**8.81**	16.55	**15.13**	**9.16**	**8.81**
	40	13	7.58	7.52	15.6	8.76	8.9	10.55	6.34	6.15	10.97	6.32	6.28	9.35	**5.53**	**5.41**	13.02	**9.33**	**5.53**	**5.41**
	60	10.7	6.46	6.03	13.2	7.76	7.6	9.07	5.26	5.08	9.04	5.06	5.12	7.62	**4.36**	4.48	11.82	**7.59**	**4.36**	**4.46**
	80	10.3	6.33	6.69	10.9	6.44	6.73	7.95	4.54	4.78	8.05	4.38	4.50	**6.96**	**3.79**	**4.00**	10.95	**6.96**	**3.79**	**4.00**
	100	10.7	6.79	6.4	9.6	5.41	5.53	7.07	4.06	4.07	7.24	4.19	3.99	**6.35**	**3.56**	**3.43**	10.32	**6.35**	**3.56**	**3.43**
	120	10.2	6.22	7.12	8.21	5.1	4.86	6.66	3.74	3.82	6.83	3.90	3.89	6.23	**3.47**	3.63	10.37	**6.13**	**3.47**	**3.58**
12	20	40.3	32.1	32	38.5	30.6	30.4	32.43	27.42	**27.14**	32.89	27.68	27.29	**32.31**	**27.28**	**27.14**	N/A	**32.31**	**27.28**	**27.14**
	40	25.9	23.4	23.4	29.2	25.7	25.5	24.94	22.87	22.79	25.00	22.61	22.53	24.27	22.60	**22.39**		**24.25**	**22.52**	**22.39**
	60	11.1	7.11	7.08	15.1	8.67	8.89	9.85	5.75	5.71	10.21	5.46	5.68	8.22	**4.83**	4.87		**8.20**	**4.83**	**4.86**
	80	12.7	9.71	9.94	14.4	10.7	10.5	10.98	8.23	8.14	11.34	**7.92**	8.00	10.40	8.10	7.96		**10.39**	**7.92**	**7.85**
	100	13.7	11.4	11.2	14.5	12.1	11.8	11.67	9.86	9.84	11.53	9.89	**9.81**	11.33	9.72	9.86		**11.18**	**9.50**	**9.81**
	120	10.8	7.05	7.01	10.1	6.14	6.09	7.29	4.18	4.30	7.78	4.29	4.33	6.87	3.72	**3.58**		**6.82**	**3.71**	**3.58**
Average		12.12	8.75	8.72	10.89	7.84	7.82	9.07	6.61	6.54	9.04	6.52	6.50	8.37	6.21	6.19	9.12	**8.33**	**6.18**	**6.17**

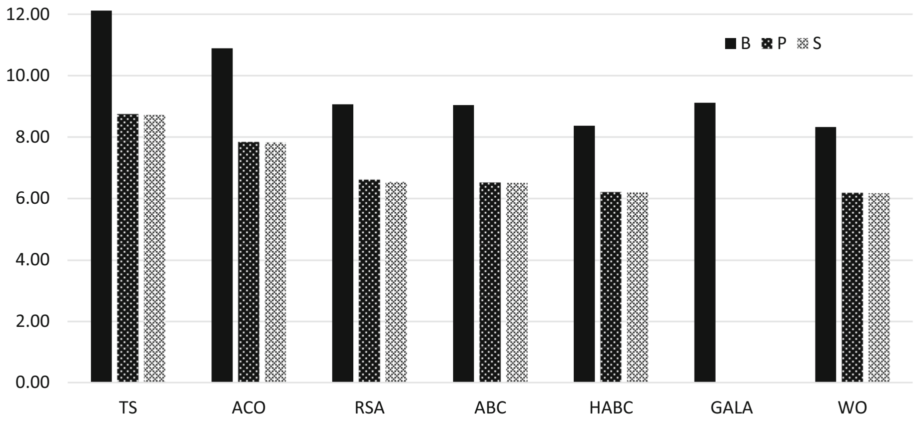

Fig. 2 Average results for all algorithms

The algorithms were evaluated by running 15 instances for each job-machine combination and under three dominance settings: balanced setup and processing times, dominant setup times, and dominant processing times; i.e. a total of 45 instances for each job-machine combination.

The algorithms are compared based on the percentage deviation from the lower bound (LB) proposed by Al-Salem (2004), using Eq. (9). Similar computational tests were conducted by Lin and Ying (2014).

$$\frac{Cmax_{Algorithm} - LB}{LB} \times 100\% \tag{9}$$

Table 8 lists the average deviations for every algorithm and all test instances, with the smallest values across algorithms bolded. It clearly indicates that WO and HABC performed the best under the three dominance settings, with the former slightly better. Figure 2 depicts the averages of the seven algorithms' deviations, and consistently with Table 8, shows an outperformance of WO and HABS, followed by ABC, RSA, GALA, ACO, and TS, respectively. GALA results were only reported under balanced dominance and up to 10 machines, as per their paper.

Tables 9, 10 and 11 provide more detailed results, by listing the mean values of the minimum, maximum, average and standard deviation of C_{max} for the 15 instances of the job-machine combinations. A similar analysis was conducted by Lin and Ying (2014) and it is repeated here for benchmarking purposes. Like Table 8, the detailed results indicate the superiority of WO and HABS, and reflect their comparable performance. Subsequently, and as WO and HABS reported the same results in many instances, paired t-tests on the average deviations from LB with 95% confidence interval were performed to verify WO's effectiveness. The results are shown in Table 12 for the three dominance settings and they confirm the statistical significance of the mean difference between WO and HABS, with the former having the lower mean.

The computational times of ACO, RSA, ABC, HABC and WO were indirectly compared using the same approach shown in Arnaout et al. (2017). In particular, and as ACO was implemented on Intel Pentium 4 @ 2 GHz, RSA on Intel Pentium 4 @ 1.5 GHz, ABC/HABC on an Intel Core Duo @ 2.66 GHz, and WO on Intel Core i3-370 M @ 2.4 GHz, direct comparison becomes meaningless. Subsequently, the different processors' CPU performance were normalized using Table 13, which is extracted from https://www.cpubenchmark.net/.

Table 9 Mean values of the minimum, average, maximum, and standard deviation of C_{max} for the Processing Dominant Times

m	*n*	TS				ACO				WO			
		Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD
2	20	1957	2010	2075	35.25	**1920**	1983	2047	28.56	**1920**	**1981**	**2045**	34.91
	40	3904	3981	4036	32.87	3832	3903	3935	29.70	**3829**	**3898**	**3927**	28.39
	60	5848	5993	6127	66.68	**5758**	5830	**5948**	48.24	**5758**	**5829**	**5948**	48.24
	80	7863	7957	8078	50.41	**7675**	**7741**	**7860**	55.12	**7675**	**7741**	**7860**	55.16
6	20	740	750	761	7.00	**731**	750	759	7.79	**731**	**745**	**757**	6.41
	40	1312	1332	1357	12.75	**1276**	1324	1344	17.17	**1276**	**1297**	**1312**	9.43
	60	1898	1939	1970	25.16	1894	1916	1934	11.66	**1870**	**1882**	**1901**	10.29
	80	2583	2620	2658	19.94	2571	2597	2622	14.98	**2541**	**2553**	**2564**	6.32
12	20	388	396	404	4.51	384	391	398	4.37	**377**	**382**	**388**	2.92
	40	732	736	742	2.39	742	750	765	6.57	**726**	**731**	**735**	2.72
	60	946	957	974	8.17	955	971	989	9.54	**930**	**937**	**942**	4.02
	80	1298	1307	1324	7.99	1304	1318	1336	9.65	**1282**	**1286**	**1292**	3.11
m	*n*	RSA				ABC				HABC			
		Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD
2	20	**1920**	1984	2051	30.06	**1920**	1982	2048	35.35	**1920**	**1981**	**2045**	34.91
	40	3841	3909	3946	29.73	3834	3906	3948	30.74	**3829**	3900	3929	28.08
	60	5777	5860	5966	48.58	5775	5850	5966	48.66	5776	5842	5966	48.04
	80	7704	7780	7900	64.05	7698	7784	7914	60.92	7691	7764	7888	59.78
6	20	**731**	746	758	6.65	**731**	**745**	**757**	6.41	**731**	**745**	**757**	6.41
	40	1292	1303	1325	9.93	1288	1302	1315	8.78	1288	1299	**1312**	8.30
	60	1879	1891	1910	9.04	1873	1890	1921	13.30	**1870**	**1882**	**1901**	10.29
	80	2548	2560	2577	8.30	**2541**	2559	2575	8.86	2543	2556	2569	6.96
12	20	**377**	**382**	**388**	3.28	**377**	383	389	3.83	**377**	**382**	**388**	2.92
	40	729	733	741	2.86	**726**	732	737	3.13	728	732	736	2.56
	60	935	945	952	4.96	933	943	950	4.35	**930**	**937**	**942**	4.02
	80	1284	1289	1300	4.42	**1282**	**1286**	**1292**	3.11	1285	1288	1293	2.57

Bold refers to smallest values across the algorithms

The CPU mark refers to the performance of the processor, where a higher number refers to better performance. The ABC/HABC platform was used as the base (factor of 1), and the running times of ACO, RSA, and WO are multiplied by their respective factors in order to have a sense of how the three algorithms measure up. The normalized computational times of the algorithms are shown in Table 14, which shows that WO computational time is within acceptable range relative to the other algorithms.

Table 10 Mean values of the minimum, average, maximum, and standard deviation of C_{max} for the balanced times

m	n	TS				ACO				GALA				WO			
		Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD
2	20	1207	1265	1325	37.57	**1192**	1238	1295	30.16	1196	1236	1297	31.39	**1192**	**1235**	**1295**	31.87
	40	2413	2487	2566	39.54	2332	2398	2478	34.87	2356	2426	2501	36.01	**2328**	**2397**	**2477**	35.03
	60	3598	3736	3818	55.61	**3509**	3575	**3621**	32.62	3580	3642	3699	38.45	**3509**	**3574**	**3621**	32.75
	80	4817	4942	5074	70.36	**4643**	**4730**	**4863**	57.80	4760	4834	4957	53.84	**4643**	**4730**	**4863**	57.80
6	20	441	449	461	6.23	441	453	462	6.77	434	446	452	4.76	**434**	**446**	**452**	4.84
	40	788	804	821	10.51	787	805	823	10.39	778	791	805	8.42	**765**	**778**	**792**	7.86
	60	1145	1179	1220	22.06	1139	1163	1192	15.87	1147	1162	1176	8.62	**1124**	**1133**	**1147**	6.20
	80	1532	1569	1616	25.83	1523	1545	1560	11.49	1526	1544	1569	10.00	**1491**	**1513**	**1531**	10.94
10	20	246	260	273	8.402	242	253	265	6.198	239	246	251	4.18	**237**	**243**	**246**	2.31
	40	466	475	483	4.881	471	486	503	8.991	469	475	481	3.48	**452**	**459**	**465**	4.33
	60	677	693	705	8.396	690	708	724	8.439	689	699	711	5.73	**666**	**673**	**681**	4.45
	80	899	921	965	15.84	908	926	946	10.28	921	927	935	4.70	**885**	**893**	**902**	4.77
12	20	236	245	254	6.65	229	242	253	5.55	N/A				**225**	**231**	**234**	2.56
	40	430	437	444	3.94	437	448	457	6.32					**427**	**431**	**435**	2.40
	60	573	577	584	3.54	586	597	613	8.93					**556**	**562**	**565**	2.60
	80	761	778	795	9.50	778	790	801	6.62					**754**	**762**	**770**	4.20

m	n	RSA				ABC				HABC			
		Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD
2	20	1193	1239	1297	29.84	**1192**	1236	**1295**	31.58	**1192**	**1235**	**1295**	31.87
	40	2362	2411	2482	32.53	2331	2405	2478	35.04	**2328**	2400	**2477**	34.53
	60	3536	3599	3647	35.79	3539	3592	3646	34.87	3523	3584	3630	35.42
	80	4702	4776	4902	54.43	4683	4764	4903	58.46	4677	4755	4884	56.65

Table 10 continued

m	n	RSA				ABC				HABC			
		Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD
6	20	438	**446**	**452**	4.21	**434**	**446**	**452**	4.81	**434**	**446**	**452**	4.84
	40	771	784	799	8.72	771	783	793	6.52	**765**	779	**792**	8.51
	60	1127	1141	1158	8.15	1125	1138	**1147**	6.34	**1124**	**1133**	1151	7.02
	80	1503	1528	1548	11.47	1500	1523	1544	12.71	**1491**	1514	1532	10.66
10	20	239	244	250	3.16	241	246	251	3.31	**237**	**243**	**246**	2.31
	40	457	464	471	3.71	457	466	478	5.15	**452**	**459**	**465**	4.42
	60	673	682	691	4.95	675	682	687	3.78	**666**	**673**	**681**	4.54
	80	891	902	912	5.25	891	902	915	6.78	**885**	**893**	**902**	4.77
12	20	**225**	**231**	236	2.83	**225**	232	237	3.18	**225**	**231**	**234**	2.56
	40	430	433	440	2.45	430	434	440	2.94	**427**	**431**	**435**	2.39
	60	564	570	575	3.24	563	572	579	4.97	**556**	**562**	**565**	2.64
	80	756	766	778	5.31	763	769	777	4.35	**754**	**762**	**770**	4.20

Bold refers to smallest values across the algorithms

Table 11 Mean values of the minimum, average, maximum, and standard deviation of C_{max} for the setup dominant times

m	n	TS				ACO				WO			
		Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD
2	20	1944	2017	2075	32.80	**1923**	1986	2047	20.62	**1923**	**1982**	**2045**	25.73
	40	3902	3985	4083	42.29	**3837**	3898	3996	37.78	**3837**	**3892**	**3971**	33.41
	60	5914	5972	6027	36.31	5780	5824	**5878**	32.47	**5773**	**5823**	**5878**	33.62
	80	7845	7939	8078	68.74	**7631**	**7715**	**7822**	51.45	**7631**	**7715**	**7822**	51.45
6	20	739	749	761	6.13	739	749	759	6.55	**734**	**744**	**752**	5.34
	40	1316	1336	1349	8.88	1310	1328	1340	7.45	**1290**	**1303**	**1314**	7.51
	60	1911	1932	1958	18.31	1884	1913	1939	15.01	**1873**	**1884**	**1898**	8.19
	80	2572	2611	2652	20.20	2548	2591	2609	15.02	**2535**	**2549**	**2565**	9.63
12	20	386	395	407	7.37	380	390	408	7.07	**375**	**381**	**384**	2.38
	40	733	737	744	3.21	744	750	759	4.03	**727**	**731**	**735**	2.40
	60	944	957	976	7.82	961	973	988	8.89	**930**	**937**	**944**	3.83
	80	1301	1310	1333	8.06	1304	1317	1334	7.71	**1275**	**1285**	**1293**	4.36
m	n	RSA				ABC				HABC			
		Min	Avg	Max	SD	Min	Avg	Max	SD	Min	Avg	Max	SD
2	20	**1923**	1985	2045	13.01	**1923**	1983	2048	26.08	**1923**	**1982**	**2045**	25.73
	40	3850	3899	3977	32.68	3848	3899	**3971**	33.07	**3837**	3894	3976	34.60
	60	5783	5848	5912	41.56	5790	5841	5885	35.61	**5773**	5832	**5878**	34.79
	80	7644	7744	7864	55.59	7653	7748	7861	55.93	7648	7738	7853	55.04
6	20	**734**	**744**	**752**	5.34	**734**	**744**	**752**	5.34	**734**	**744**	**752**	5.34
	40	1293	1307	1320	7.72	**1290**	1307	1320	7.29	1294	1304	1315	7.00
	60	1877	1892	1910	8.29	1875	1889	1912	11.34	**1873**	1885	**1898**	7.79
	80	2545	2559	2583	11.10	2537	2555	2570	11.14	**2535**	2552	2584	13.77
12	20	**375**	**381**	**384**	2.38	**375**	**381**	386	2.76	**375**	**381**	**384**	2.38
	40	728	733	737	2.48	729	732	**735**	1.99	**727**	**731**	**735**	2.40
	60	940	945	953	3.82	941	945	950	2.58	**930**	938	945	3.96
	80	**1275**	1289	1299	5.26	1277	1287	1296	4.08	1279	1287	**1293**	3.58

Bold refers to smallest values across the algorithms

Table 12 Paired t-tests on deviations from LB

Paired t-tests	Dominance	Mean DIFFERENCE (MD)	SD	t-stat	Two-tailed p	95% CI on MD
HABC–WO	*P*	0.051193	0.080877	3.797855	0.000557876	[0.02383, 0.07502]
	B	0.0936333	0.15233	3.688036	0.000761698	[0.04209, 0.14517]
	S	0.0405932	0.070107	3.474117	0.001384536	[0.01687, 0.06431]

Table 13 Normalization of CPU performance

	ACO Intel Pentium 4 @ 2.00 GHz	RSA Intel Pentium 4 @ 1.5 GHz	ABC HABC Intel Core Duo @ 2.66 GHz	WO Intel Core i3-370 M @ 2.4 GHz
CPU mark	189	131	1719	2025
Normalization factor	0.1099	0.0762	1	1.1780

Table 14 Normalized computational times (sec)

m	n	ACO	RSA	ABC	HABC	WO
2	20	7.06	0.16	0.9	1.4	1.77
	40	14	0.79	9.12	2.8	13.8
	60	28.1	2.41	31.4	4.2	53.4
	80	45.8	4.62	86.4	5.6	82.9
	100	68.9	8.11	178	7	177
	120	96.1	13.9	317	8.41	20.1
4	20	6.73	0.15	0.52	2.81	0.46
	40	10.6	0.84	5.56	5.6	6.04
	60	21.2	2.55	21.1	8.41	17.9
	80	34.2	5.22	50.4	11.2	44.8
	100	61.3	9.21	118	14	124
	120	80	17.5	200	16.8	152
6	20	6.6	0.17	23.8	22.5	1.16
	40	10.4	1.02	4.02	8.41	5.34
	60	20.7	2.48	15	12.6	9.37
	80	33.4	5.12	32.2	16.8	17.7
	100	59.9	11	76	21	53.9
	120	82.8	17.7	143	25.2	114
8	20	7.26	0.23	0.38	5.6	1.47
	40	11.3	0.98	3.07	11.2	2.07
	60	22.7	2.59	9.89	16.8	9.27
	80	36.6	6.64	28.2	22.4	9.13
	100	65.6	13.4	28	28	38.4
	120	90.8	20.1	110	33.6	55.8
10	20	7.45	0.25	0.31	7	3.92
	40	11.7	1.11	1.92	14	5.05
	60	23.4	3.42	8.16	21	12.1
	80	37.7	7.71	22.5	28	10.5
	100	67.6	16.8	44.3	35	60.5
	120	93.5	30.2	82.9	42	92
12	20	N/A	0.28	0.44	8.41	6.07
	40		1.05	1.7	16.8	2.72
	60		3.92	7.29	25.2	7.86
	80		9.01	20	33.6	12.9
	100		13.2	34.8	42	46.4
	120		31.7	62.5	50.4	54.8
Average	38.8	7.38	49.4	17.7	36.8	

4 Conclusion and future research

In this paper, we have introduced a Worm Optimization algorithm (WO) for minimizing the makespan on the unrelated parallel machine scheduling problem with sequence-dependent setup times.

WO was compared to tabu search (TS) by Helal et al. (2006), ant colony optimization (ACO) by Arnaout et al. (2010), restrictive simulated annealing (RSA) by Ying et al. (2012), genetic algorithm (GA) by Eroglu et al. (2014), and ABC/HABC by Lin and Ying (2014). The tests showed the superiority of WO, followed by HABC, ABC, RSA, GALA, ACO, and TS last. WO's average deviation from the lower bound was less than 1% for some job-machine combinations, with an average of ~ 7% over all dominance settings. In addition, the prediction expression that was generated using Design of Experiments indicated that a lower production rate ϕ leads to better results, as this will delay the convergence of WO. Finally, the worms' unique behaviors were attributed to better solutions; e.g. the solitary behavior ensures that more search space is visited, and the Dauer behavior gives a unique convergence to WO.

An extension to this research would be to test WO on the stochastic version of the problem, in particular, under machine breakdowns.

Acknowledgements This work was funded in part by a Grant from the Kuwait Foundation for the Advancement of Sciences (KFAS Grant # P115-18EO-02). The authors would like to express their sincere gratitude to Reviewer 1 for his detailed and constructive comments, which aided in significantly improving the quality of this work.

References

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research, 246,* 345–378.

Al-Salem, A. (2004). Scheduling to minimize makespan on unequal parallel machines with sequence dependent setup times. *Engineering Journal of the University of Qatar, 17,* 177–187.

Arnaout, J.-P. (2016). Worm optimization for the traveling salesman problem. In G. Rabadi (Ed.), *Heuristics, meta-heuristics and approximate methods in planning and scheduling, international series in operations research & management science*. Switzerland: Springer.

Arnaout, J.-P. (2017). Worm optimization for the multiple level warehouse layout problem. *Annals of Operations Research*, *269*, 29–51.

Arnaout, J.-P., ElKhoury, C., Karayaz, G. (2017). Solving the multiple level warehouse layout problem using ant colony. *Operational Research: An International Journal*, 1–18.

Arnaout, J.-P., Musa, R., & Rabadi, G. (2014). A two-stage Ant Colony optimization algorithm to minimize the makespan on unrelated parallel machines: Part II—enhancements and experimentations. *Journal of Intelligent Manufacturing, 25,* 43–53.

Arnaout, J.-P., Rabadi, G., & Musa, R. (2010). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing, 21,* 693–701.

Avalos-Rosales, O., Angel-Bello, F., & Alvarez, A. (2015). Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *The International Journal of Advanced Manufacturing Technology, 76,* 1705–1718.

Brabazon, A., McGarraghy, S. (2018). Worm foraging algorithm. In G. Rozenberg, Th. Bäck, A. E. Eiben, J. N. Kok, & H. P. Spaink (Eds.), *Foraging-inspired optimisation algorithms. Natural computing series*. Springer, Cham.

Chang, P. C., & Chen, S. H. (2011). Integrating dominance properties with genetic algorithms for parallel machine scheduling problems with setup times. *Applied Soft Computing, 11,* 1263–1274.

Diana, R. O. M., de França Filho, M. F., de Souza, S. R., & de Almeida Vitor, J. F. (2015). An immune-inspired algorithm for an unrelated parallel machines' scheduling problem with sequence and machine dependent setup-times for makespan minimisation. *Neurocomputing, 163,* 94–105.

Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *BioSystems, 43,* 73–81.

Eroglu, D. Y., Ozmutlu, H. C., & Ozmutlu, S. (2014). Genetic algorithm with local search for the unrelated parallel machine scheduling problem with sequence-dependent set-up times. *International Journal of Production Research, 52*(19), 5841–5856.

Ezugwu, A. E., Adeleke, O. J., & Viriri, S. (2018). Symbiotic organisms search algorithm for the unrelated parallel machines scheduling with sequence-dependent setup times. *PLoS ONE, 13*(7), 1–23.

Fisher, R. A. (1960). *The design of experiments*. New York: Hafner Publishing Company.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco: W. H. Freeman and Company.

Helal, M., Rabadi, G., & Al-Salem, A. (2006). A Tabu search algorithm to minimize the makespan for unrelated parallel machines scheduling problem with setup times. *International Journal of Operations Research, 3*(3), 182–192.

Karp, R. M. (1972). Reducibility among combinatorial problems. In R. E. Miller & J. W. Tatcher (Eds.), *Complexity of computer computations* (pp. 85–103). New York: Plenum Press.

Lin, S.-W., & Ying, K.-C. (2014). ABC-based manufacturing scheduling for unrelated parallel machines with machine-dependent and job sequence-dependent setup times. *Computers & Operations Research, 51,* 172–181.

NIST/SEMATECH e-handbook of statistical methods. http://www.itl.nist.gov/div898/handbook/. Accessed February, 2018.

Rabadi, G., Moraga, R., & Al-Salem, A. (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing, 17,* 85–97.

Ross, P. (1996). *Taguchi techniques for quality engineering*. New York: McGraw Hill.

SchedulingResearch. (2005). http://SchedulingResearch.com. Accessed February, 2018.

Taguchi, G. (1993). *Taguchi methods: Design of experiments*. Michigan: American Supplier Institute Inc.

Wang, L., Wang, S., & Zheng, X. (2016). Hybrid estimation of distribution algorithm for unrelated parallel machine scheduling with sequence-dependent setup times. *IEEE/CAA Journal of Automatica Sinica, 3*(3), 235–246.

Ying, K.-C., Lee, Z.-J., & Lin, S.-W. (2012). Makespan minimization for scheduling unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing, 23*(5), 1795–1803.