A New Foraging-Based Algorithm for Online Scheduling

Koen van der Blom
LIACS, Leiden University
Leiden, The Netherlands
k.van.der.blom@liacs.leidenuniv.nl

Thomas Bäck
LIACS, Leiden University
Leiden, The Netherlands
t.h.w.baeck@liacs.leidenuniv.nl

ABSTRACT

While much work exists on scheduling, literature in the subfield of online scheduling remains sparse. As with many problems, online scheduling has parallels with natural phenomena. Specifically, online scheduling can be seen in the division of labour among colony insects, such as ants. Although multiple different biological models exist for division of labour, the only one to have been applied in online scheduling is the reinforced threshold model, for instance in the form of the ant task allocation (ATA) algorithm. However, it is neither known how it compares to other models, nor in which applications any of them excel. This paper studies the foraging for work (FFW) model as a possible alternative. To do so, an algorithmic description of the FFW model is introduced, and it is compared with the ATA algorithm on the truck painting problem. For this problem, tasks of various types are scheduled in a flowshop with multiple identical machines in an online fashion. FFW is shown to be very effective at minimising setup time, which is incurred when switching to tasks of different types. Furthermore, this allows FFW to outperform the threshold based approaches when the scheduling environment is placed under heavy load.

CCS CONCEPTS

• **Theory of computation** → *Scheduling algorithms*; *Online algorithms*; • **Computing methodologies** → *Planning under uncertainty*; *Multi-agent planning*; *Bio-inspired approaches*;

KEYWORDS

Online Scheduling, Division of Labour, Swarm Intelligence, Colony Insects, Flowshop

ACM Reference Format:
Koen van der Blom and Thomas Bäck. 2018. A New Foraging-Based Algorithm for Online Scheduling. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205496

1 INTRODUCTION

In [23] the case is made that, although scheduling as a whole is quite a mature field, online (or: dynamic) scheduling remains poorly studied. Further, they identified agent based approaches, and specifically those inspired by nature, as an area that shows potential for online scheduling tasks. More recently, [19] reviewed the state-of-the-art in online scheduling, and also identified multi-agent systems as a promising subfield in online scheduling. Multiple agent based systems already exist, some based on market principles [3, 14–16], and others based on colony insect behaviour [3, 5]. However, these too have only seen limited development.

Morely et al. [14–16] introduced ideas based on market behaviour and chaos theory into online scheduling. Essentially, they let machines compete for the available jobs. Both [3] and [5] noted the similarity between the scheduling mechanism used by [14–16], and the division of labour among colony insects in nature. Subsequently, both groups developed scheduling algorithms based on the reinforced threshold model described by [25], and outperformed the market based approach. In short, the model assumes insects will perform jobs based on whether their threshold for this job is reached or not. Subsequently, these thresholds are updated based on which jobs they perform, and which jobs are available.

Multiple extensions to these initial two algorithms were proposed. In [12] parameter tuning is investigated for the algorithm by [3], as well as various tie-breaking rules, and different local and global update schemes for thresholds. However, they do not propose a new algorithm. The algorithm by [5] was extended by [17, 18], who considered a number of changes to improve performance, resulting in the ant task allocation (ATA) algorithm. A later study by [13] investigates parameter updates with fuzzy logic for the algorithm by [5]. Unfortunately, some essential details about their fuzzy system are missing, as such this algorithm is not considered here. More recently, [4] introduced a new threshold adaptation scheme and applied it to job allocation for robot swarms. Further, [11] establishes theory for why threshold models work.

Although the reinforced threshold model has seen considerable success, it is hardly the only explanation for division of labour in colony insects. The works by [1, 7] review the biological phenomena underlying colony insect division of labour, as well as the different models based on those phenomena. One such alternative, the foraging for work (FFW) model [27], is considered here.

Given that agent based approaches were identified as promising [19, 23], it is worth investigating different alternatives rather than blindly focusing on a single approach. Both the potential for a generally better approach, as well as the possibility of improving in a niche area make this a valuable effort.

This paper introduces an algorithmic description for the FFW model. Further, the resulting FFW algorithm and the existing ATA algorithm are compared on empirical experiments, along with a random assignment strategy. Moreover, the influence of parameter settings on the FFW algorithm is analysed in more detail.

The remainder of this paper is structured as follows. Section 2 introduces the truck painting problem. Followed by this, Section 3

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205496

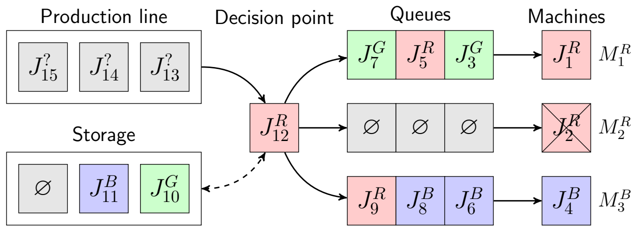

Figure 1: Truck painting problem.

describes the ant task allocation and foraging for work algorithms. In Section 4 the experimental setup is discussed, which naturally leads to the results presented in Section 5. Finally, the paper is summarised and concluded in Section 6.

2 PROBLEM

Truck painting as an online scheduling problem for a flowshop with identical parallel machines was first introduced by Morley et al. [14–16] in the early nineties. Since then, this has become a common test problem in online scheduling. As such, the truck painting problem is also used here, and will be briefly introduced in the following. In this paper general terminology is favoured over problem specific terminology. I.e. jobs rather than trucks, types rather than colours, and processing rather than painting.

The truck painting problem (TPP) is visualised in Figure 1. Jobs (trucks) coming off the production line have to be processed (painted). However, the order in which they are completed is unknown. As such, the required processing type (colour) is only known once they arrive in the processing facility (paintshop), indicated by the decision point in Figure 1. Due to this, the processing schedule cannot be optimised in advance, and has to be constructed on the fly. When the job arrives, and the required type becomes apparent, it is assigned to a machine and added to its FIFO (first in, first out) queue. Once in the queue, jobs can neither be reassigned to another machine, nor can they change position with other jobs in the same queue. In case the system overflows (all queues are full), or the scheduling mechanism delays the assignment decision, jobs can be placed in temporary storage, where they await later assignment. When jobs of different types are placed in the same queue a machine has to switch job types. Switching to a different type (performing a setup) incurs a cost. In case of painting machines this process involves flushing away paint of the previous colour, which takes both time, and results in a loss of paint (a financial cost). It should be noted that these can be significant costs, [15] reported reducing the required number of setups by half, and saving nearly three million USD per year as a result. Finally, machines can also break down during operation. Naturally, when a machine is broken it cannot process jobs until it is repaired.

3 ALGORITHMS

To solve the truck painting problem, two algorithms are considered as described in detail in the following subsections. First, the ant task allocation (ATA) algorithm, as previously introduced in [17, 18]. Second, the foraging for work (FFW) algorithm introduced in this paper, based on the FFW model from [27].

Note that here a distinction is made between *model* and *algorithm*. *Model* is used to refer to an approximating description of natural phenomena, whereas *algorithm* denotes the description of a computational program, possibly based on a biological model. So the FFW model and the FFW algorithm are distinct concepts here.

3.1 Ant Task Allocation

Inspired by the fixed threshold model [2] and the later reinforced threshold model [25], an insect inspired algorithm was proposed by [3]. The idea behind both the threshold model and the algorithm based thereon is that individual insects have a, possibly variable, threshold for every job type. When the stimulus for a job reaches this threshold the insect will perform jobs of that type. The stimulus is the perceived need for a job through interaction with the environment and other colony members. E.g., often running into trash may incite nest cleaning, while rarely encountering trash likely results in sticking with the current job. This threshold is influenced by a number of factors, such as genetics and experience [6].

Another insect inspired algorithm, called R-Wasps, was proposed by [5]. This was based on the same reinforced threshold model by [2] and [25]. Differences between the two come from the manner in which the concept of thresholds is translated into an algorithm. Where [3] used the threshold to determine the height of bids made by machines, [5] use the threshold in a more insect-like manner to determine whether or not a machine will compete for a certain job at all. In situations with multiple bidders, this is followed by a wasp-inspired dominance contest, based on work by [24, 26], to determine which machine acquires the job.

Ant task allocation (ATA) [17, 18] considers a number of extensions to the R-Wasps algorithm [5]. Since there are differences between the descriptions of the algorithm in [17] and [18] (and no clear preference is indicated), a description of the implementation used here is included in Algorithm 1. Assignment and comparison are denoted with =, whereas variable updates are denoted with :=.

Every minute (here: every fifth timestep) a job is released. Following this, all unassigned jobs are put up for auction in order of their release time (new jobs first, then any waiting in storage; line 4). Given an available job, ATA first considers for each machine whether or not it will compete for this job according to Equation 1 (line 6). The probability to bid depends on the threshold $\theta_{w,j}$ of machine w for a job of type j, and the stimulus S_i emitted by a job i that is on offer. In [17] the stimulus is only mentioned as being "equal to the length of time the job is already waiting", but no time unit is given. Here the stimulus is initialised as one, and incremented by one for every second it remains unassigned. Further, the threshold is bound in $[\theta_{min}, \theta_{max}]$.

$$P_w(bid|\theta_{w,j}, S_i) = \frac{S_i^2}{S_i^2 + \theta_{w,j}^2} \tag{1}$$

Every machine that bids (its probability to bid is greater than a uniformly at random drawn number from [0.0, 1.0]), does so with a certain force F_w (line 8). This is computed according to Equation 2, where T_w^{proc} and T_w^{setup} are respectively the processing and setup time required for all jobs currently in the machine's queue. The term T_w^{new} indicates any setup time that may be needed between the last job in queue, and the new job on offer.

Algorithm 1 Ant Task Allocation (ATA)

```
Require: $\gamma_1, \gamma_2, \delta_1, \delta_2, \delta_3, \theta_{min}, \theta_{max}$
 1: $T = 0$
 2: while $T < maxT$ do
 3:     if T % 60 = 0 then
 4:         for every unassigned job $i$, of type $j$
 5:         for all machines $w \in \{1, \dots, m\}$ do
 6:             $P_w(bid|\theta_{w,j}, S_i) = S_i^2/(S_i^2 + \theta_{w,j}^2)$                ▷ Eq. 1
 7:             if $U(0,1) < P_w$ then        ▷ uniform random number
 8:                 $F_w = 1.0 + T_w^{proc} + T_w^{setup} + T_w^{new}$       ▷ Eq. 2
 9:             else
10:                 $F_w = 0$
11:                 $\theta_{w,j} := \theta_{w,j} - \gamma_2$
12:             end if
13:         end for
14:         if $\forall_{w \in \{1,...,m\}} : F_w = 0$ then                ▷ no bids placed
15:             $\theta_{w,j} := \theta_{w,j} - \gamma_1$
16:         else
17:             $P_w(F_1, ..., F_m) = \frac{1}{F_w^2} / \sum_{i=1}^{m} \frac{1}{F_i^2}$                ▷ Eq. 5
18:             select the winner by proportional selection
19:             add job $i$ to the queue of the winner
20:         end if
21:     end if
22:     for all machines $M_w, w \in \{1, \dots, m\}$ do
23:         for all job types $j$ do         ▷ $A$ is active set, $I$ is idle set
24:             if $M_w \in A \subset M \wedge j = type(Q_w^{tail})$ then
25:                 $\theta_{w,j} := \theta_{w,j} - \delta_1$
26:             else if $M_w \in A \subset M$ then
27:                 $\theta_{w,j} := \theta_{w,j} + \delta_2$
28:             else if $M_w \in I \subset M$ then
29:                 $\theta_{w,j} := \theta_{w,j} - \delta_3^t$
30:             end if
31:             adjust $\theta_{w,j}$ to bound if $[\theta_{min}, \theta_{max}]$ is exceeded
32:         end for
33:     end for
34:     for all unassigned jobs $i$ do
35:         $S_i := S_i + 12$
36:     end for
37:     $T := T + 12$
38: end while
```

$$F_w = 1.0 + T_w^{proc} + T_w^{setup} + T_w^{new} \tag{2}$$

Machines that do not bid have their threshold for the job type on offer reduced by γ_2 according to Equation 3 (line 11).

$$\theta'_{w,j} = \theta_{w,j} - \gamma_2 \tag{3}$$

If none of the machines bid on the job in question, all machines reduce their threshold for this job type by γ_1 according to Equation 4 (line 15).

$$\theta'_{w,j} = \theta_{w,j} - \gamma_1 \tag{4}$$

Otherwise, the probability to win P_w is found according to Equation 5, for every machine $w \in \{1, \dots, m\}$ (line 17).

$$P_w(F_1, ..., F_m) = \frac{\frac{1}{F_w^2}}{\sum_{i=1}^{m} \frac{1}{F_i^2}} \tag{5}$$

Furthermore, on every time step (five per minute) all threshold values are updated as follows. All active machines $M_w \in A \subset M$ (either processing a job, or setting up for the next job type) reduce the threshold for the job type that is last in queue Q_w^{tail} with δ_1 as in Equation 6 (line 25).

$$\theta'_{w,j} = \theta_{w,j} - \delta_1 \tag{6}$$

For all other job types, they increase the threshold by δ_2 according to Equation 7 (line 27).

$$\theta'_{w,j} = \theta_{w,j} + \delta_2 \tag{7}$$

Idle machines $M_w \in I \subset M$ use Equation 8 to decrease their threshold for all types by δ_3^t, where the exponent t is the idle time in seconds (line 29).

$$\theta'_{w,j} = \theta_{w,j} - \delta_3^t \tag{8}$$

It should be noted that it is not exactly clear from [17, 18] how often thresholds are updated compared to the number of time steps. As such, here the five updates per time step (minute) from the R-Wasps [5] algorithm are maintained. Further it is noted that this approach is arbitrary, and one update per time step (with correctly adjusted δ parameters) could achieve the same effect.

Finally, stimuli for unassigned jobs are increased to improve their chances in the next iteration (lines 34-36).

3.2 Foraging for Work

Foraging for work (FFW) as a biological model was introduced in [27] as an explanation for the distribution of jobs among ants. Although the model received a number of critiques with regard to being too simplistic to fully explain ant behaviour in nature [21, 22, 28], this simplicity is exactly why it is of interest in an algorithmic sense. The core idea behind the model is that ants select their jobs primarily based on their location. As a result, they would favour jobs in their vicinity over those further away. Moreover, this would provide an explanation for ants of different ages favouring different jobs. After all, young ants hatch deep in the nest and naturally come across jobs such as taking care of the brood. As more and more ants hatch, work becomes scarce deep in the nest, and ants start moving away while looking for new jobs. Through this process, ants gradually move to new regions and find different types of jobs (e.g. defending the nest, or foraging for food).

To apply the concept of foraging for work in an algorithm, job types must be distributed somehow. Since this paper investigates the viability of FFW as an algorithm, a simple spatial distribution is chosen; alternatives may be studied in the future. Specifically, the different job types are placed in a circular graph such that every type has two neighbouring types. As a result, there is no start- or endpoint, which is desirable since every type is intrinsically equal. This topology is equivalent to the 'ring neighbourhood' sometimes used in Particle Swarm Optimisation. In Algorithm 2 the FFW algorithm is defined based on the concepts from the FFW model.

Every machine $M_w, w \in \{1, \ldots, m\}$ is initialised with a job type $j \in \{1, \ldots, k\}$ chosen uniformly at random, and a starting location $\ell \in \{1, \ldots, k\}$ with the same value (lines 1-4). In the main loop, first the machines are shuffled such that they are processed in a different order each iteration (line 6). Next, all active machines $A_w \in A \subset M$ check whether there is a job available for their current active type (line 8). If this is the case, the job is added to their queue (if there is space). Note that each machine checks only once, and therefore adds at most one job to their queue per iteration. All idle machines $I_w \in I \subset M$ follow the same procedure, with those that find a job becoming active (lines 13-16). By allowing active machines to do this first, fewer type switches should be necessary. Finally, machines that are still idle take the closest job available within the step size limit σ, if any such job exists (lines 19-21). When a job is available at the same distance in both directions, one is chosen at random. If no job is found the machine moves σ steps in a random direction (line 23).

Algorithm 2 Foraging for Work (FFW)

```
Require: $\sigma$                                          ▷ step size
1:  for all $M_w, w \in \{1, \ldots, m\}$ do
2:      $M_{w,j} = U(1, k)$                    ▷ discrete uniform random job type
3:      $M_{w,\ell} = M_{w,j}$                  ▷ starting location equals job type
4:  end for
5:  while runtime is not over do
6:      shuffle $M$                            ▷ $A$ = active, $I$ = inactive
7:      for all $A_w \in A \subset M$ do
8:          if job of type $A_{w,j}$ exists $\wedge$ $A_w$ has queue space then
9:              add job to the queue of $A_w$
10:         end if
11:     end for
12:     for all $I_w \in I \subset M$ do
13:         if job of type $I_{w,j}$ exists then
14:             add job to the queue of $I_w$
15:             set $I_w$ to active (removing it from $I$)
16:         end if
17:     end for
18:     for all $I_w \in I \subset M$ do
19:         $x$ = closest job to $I_{w,\ell}$    ▷ choose randomly if multiple
20:         if $x$ is within $\sigma$ distance from $I_{w,\ell}$ then
21:             move there and add job to the queue of $I_w$
22:         else
23:             move $\sigma$ steps in a random direction
24:         end if
25:     end for
26: end while
```

4 EXPERIMENTS

The problem parameters considered in [3] are used as basis for the experimental setup. Whenever there is a reason to deviate from this or something was not specified, the problem parameters as in [5] serve as second option. When neither of these make a clear statement a value is chosen that seems reasonable given the other settings. In order to measure throughput, the runtime and duration of production will deviate from [3], who measured the makespan (time required to complete all jobs) instead. As in [5], simulation will take 1000 minutes with jobs being produced at a rate of one per minute and requiring a processing time of three minutes. This leads to a theoretical optimal throughput of 998 jobs (all jobs released from t_0 to t_{997} have a chance of completing their three minute processing time). Although, even with optimal assignment, jobs released toward the end may have to deal with setups or breakdowns, making this optimum an unrealistic target. Each job will have to be processed for one of twenty job types, selected according to probabilities specified later in this section. Every minute there is a 0.05 probability for a uniformly at random selected machine to break down for a uniformly at random selected number of minutes in $[1, \ldots, 20]$. Changing the job type of a machine requires a setup of three minutes. Campos et al. [3] considered cases with between six and fifteen machines. They mention that when using eight machines or more, storage is rarely needed, therefore eight machines will be used here. Queues with space for five jobs are used like in [3]. In addition to there being no reason to change this, changes would also invalidate the comment about eight machines. The initial type of every machine is chosen uniformly at random as in [5], since [3] did not specify how they did this. For the same reason, thresholds are also initialised as considered by [5]. The threshold belonging to the initial type of a machine is set to θ_{min}. All other thresholds are drawn uniformly at random from the interval $[\frac{\theta_{max}}{2}, \theta_{max}]$. To allow all algorithms to work as intended, there is no limit on the size of the storage space, but it is kept track of as detailed later.

Table 1: Considered Problem Scenarios

Scenario	Characteristics
1. Uniform	A standard situation with an uniform colour distribution, with parameters as described in the experimental setup
2. Non-uniform	A standard situation with a non-uniform colour distribution: one colour 70%, one colour 15%, one colour 7%, one colour 4% and the remaining sixteen colours 0.25% each
3. Overload	A heavy load situation with a non-uniform colour distribution and twice as many trucks being produced every minute

To evaluate the performance of the two algorithms they are compared on three different scenarios as detailed in Table 1. Moreover, a uniform random strategy is included in the experiments to serve as a baseline. Each algorithm is tested on each scenario for 1000 repetitions. These scenarios are the same as in [3], except the fourth scenario considered there is not used. Both the third and fourth scenario consider a heavy load situation. The algorithms were found to have difficulty with the third scenario, as such the even more extreme fourth scenario would provide little new information.

The first scenario considers a uniform distribution of job types. With all types occurring equally frequently, and eight machines versus twenty job types, this scenario especially tests how effective an algorithm is in timing type changes. In the second scenario a non-uniform job type distribution is considered. This is more realistic in the sense that it simulates a real paintshop, where more

Table 2: Performance Metrics

Metric	Goal	Description
Throughput	Max	The number of jobs completed in a given time frame
Setup time	Min	The time spent switching to different job types
Flow time	Min	The time a job spends between entering and leaving the system
Queue length	Min	The number of jobs waiting in the queue of a machine
Storage used	Min	The number of jobs awaiting assignment in storage

Table 3: Parameter Settings

Algorithm	Parameters	Origin
Random	-	-
ATA	$\delta_1 = \delta_2 = 14, \delta_3 = 1.01,$ $\gamma_1 = \gamma_2 = 50, \theta_{min} = 1, \theta_{max} = 2000$	[17]
FFW	$\sigma = 2$	Original

popular colours (job types) appear more frequently. Finally, in the third scenario a heavy load situation is considered, where twice as many jobs have to be processed. This serves to test how well the algorithms perform under stress.

Here the scenarios from [3] are considered rather than those from [17, 18]. This is because the latter consider situations where the job type distribution changes halfway through the experiment. That tests how well the system adapts, rather than how well it performs. Since a well performing system will adapt to a good thresholds quickly after starting, this is however redundant in the sense that it simply has to do so twice, rather than once.

Three approaches are then evaluated on these three scenarios: (a) ATA, (b) FFW, and (c) random assignment. Since FFW was found to be effective in some cases, it is also evaluated further on the first two scenarios with different step sizes.

For performance assessment of the different approaches a number of metrics are considered. Brief descriptions for these metrics are included in Table 2. Throughput should be maximised since completing more jobs per time unit is naturally beneficial. Minimisation of the (total) setup time is important since it both increases potential processing time, as well as reducing setup costs (e.g. monetary) involved with switching between job types. Flow time measures the time a job spends in the system, by minimising this it is ensured individual jobs do not incur significant delays. Space often comes at a premium, as such minimising the required space, both in terms of queues and storage, is also valuable.

Parameter settings used for the experiments are given in Table 3. Settings for ATA are taken from [17]. Note that these are not optimised for the experiments considered here. For FFW a step size $\sigma = 2$ is chosen. Since machines only move further than this when they go idle, this should allow machines to specialise in a subset of the available job types.

5 RESULTS

From Figure 2 it is evident that FFW is able to outperform ATA in every metric, except in used storage space. Presumably part of the reason FFW outperforms ATA here comes from delaying job assignments by using the storage area. However, the difference in storage use is too small to be the full explanation. Especially the performance in terms of total setup time is impressive, where FFW uses around a thousand minutes less compared to ATA. A disadvantage of the insect based approaches is the significant use of storage space, whereas with random assignment storage space is rarely used. However, when total space use (storage + queue) is considered, FFW is still better than random, even in the worst case.

When looking at the uniform case with different step sizes for FFW in Figure 3, it appears that there is a trade-off between the throughput and total setup time metrics. Larger step sizes result in higher throughput (although it quickly levels off), while the total setup time gets longer. The other three metrics all improve with larger step sizes. Clearly, FFW can easily be configured for different needs (minimal setup time or maximal throughput) in case of uniformly distributed job types.

In the non-uniform situation shown in Figure 4, FFW is no longer competitive in most metrics, even when compared to the random strategy. However, it maintains an impressive total setup time. Given how large the gain in setup time is, FFW can still be preferred over the random approach, since the difference in other areas is not extreme.

When the best step size from Figure 5 is used, FFW becomes even better in terms of setup time. However, it is still not competitive with ATA on the other metrics. As such, for this sort of non-uniform job type distribution the ATA algorithm would usually be preferred, except when setup time is of high importance.

From looking at the setup time in Figure 5 it is evident that a small step size is not always better for FFW, as was assumed in the experimental setup. Even so, what causes the somewhat chaotic relation between step size and setup time here is not immediately obvious. The explanation seems to be that large steps can be beneficial for machines that went idle to return to the type they were last active in. When they go back to work there, no setup is required. Further, the step size being divisible by the number of job types (here twenty) is also beneficial. This increases the likelihood to land on the last active type, making it possible to find a job there in the next time step. In this case, a step size of ten combines these factors, and as a result shows the best performance.

When the system is placed under heavy load, as in Figure 6, only FFW is able to cope with the problem. It appears the ability to minimise the time wasted on setups is essential to keeping the system operational. Only the average flow time becomes quite large with an average of twenty-five minutes, this is however true for all three approaches. Another downside is the amount of storage space required. Although, this might be alleviated by reducing the load slightly. What is surprising is that despite the job distribution remaining non-uniform, FFW now outperforms the other approaches. There appears to be a balance where given sufficiently heavy load FFW is better, but ATA is better otherwise. Further investigation is needed to find this turning point and make definite statements about which algorithm is better under which conditions.

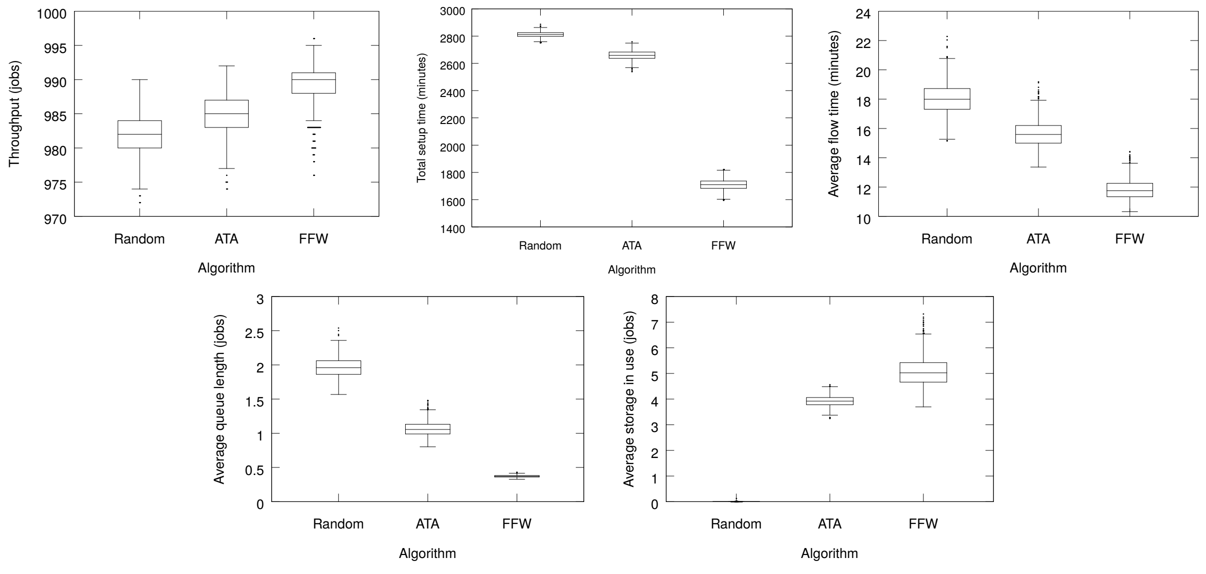

Figure 2: Results experiment 1: Uniformly distributed job types, 1000 repetitions.

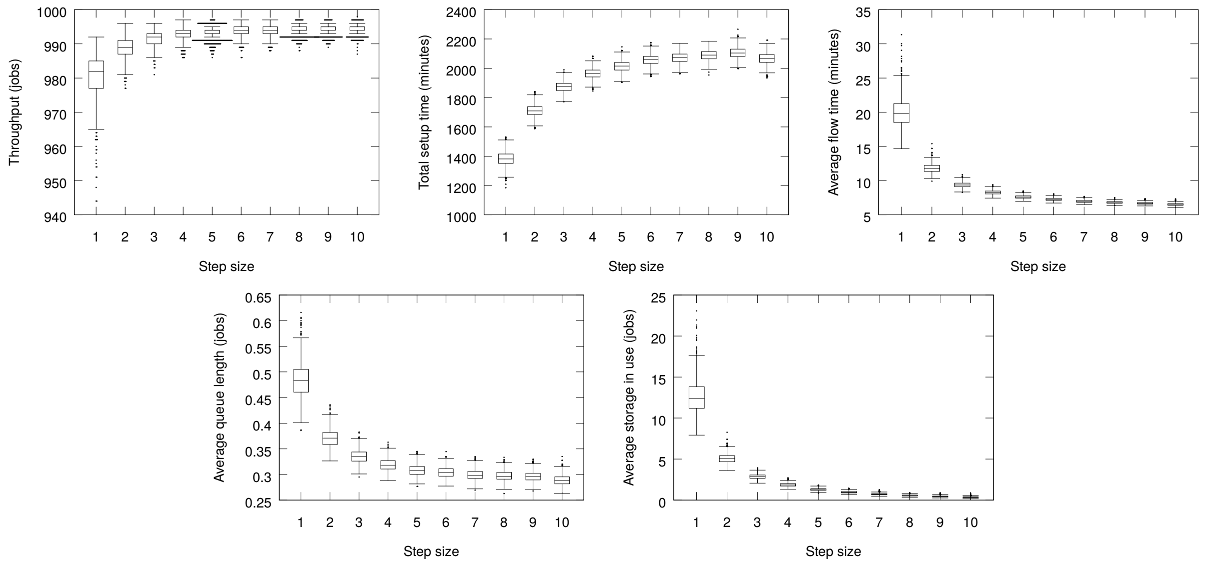

Figure 3: Results experiment 1: Uniformly distributed job types - FFW with different step sizes, 1000 repetitions.

6 CONCLUSION

In this paper two algorithms based on biological models of division of labour in colony insects have been compared: (1) the ant task allocation (ATA) algorithm [17, 18], based on the reinforced threshold model [25], and (2) the newly proposed foraging for work (FFW) algorithm, based on a model of the same name [27]. For comparison three scenarios of the truck painting problem [14–16], which is a flowshop with identical parallel machines, have been considered.

FFW showed excellent performance on the scenario with uniformly distributed job types, and on the scenario with a heavy load of non-uniformly distributed job types. However, it was not competitive on the scenario with normal load and non-uniformly distributed job types. There ATA performed better in every metric except total setup time.

While the FFW algorithm showed the best performance in multiple scenarios, this comes with the caveat that algorithm parameters were not tuned for the specific problem scenarios considered. In order to draw definite conclusions, parameter tuning is an important step for future work. It has also been shown that, due to its single parameter, FFW can be configured easily depending on the requirements of the use case. Whereas, ATA requires more effort in this regard, with seven parameters that need to be set.

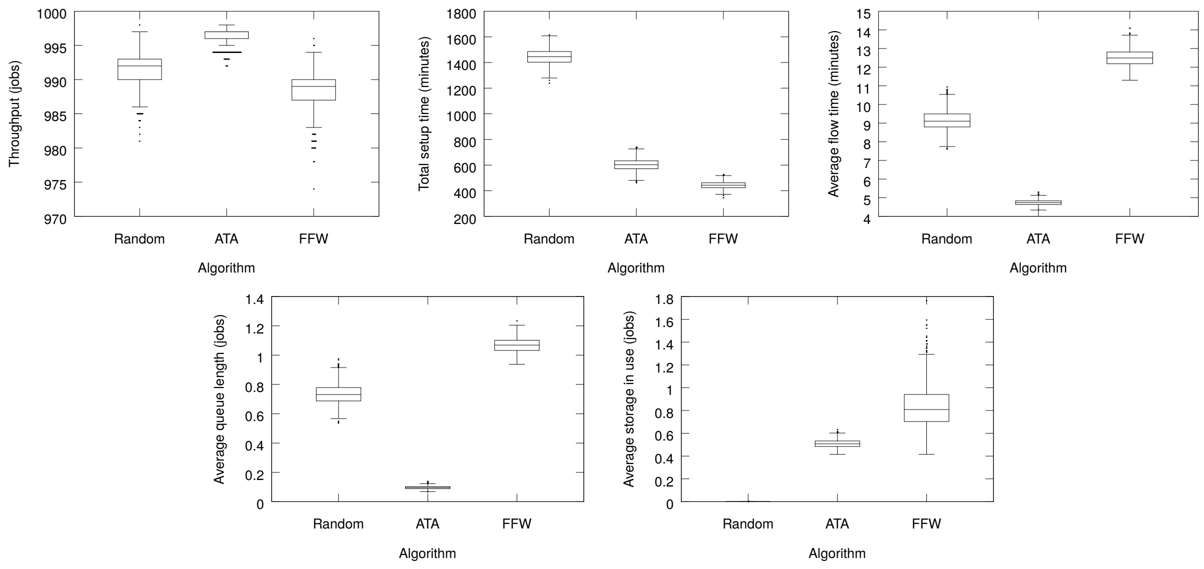

Figure 4: Results experiment 2: Non-uniformly distributed job types, 1000 repetitions.

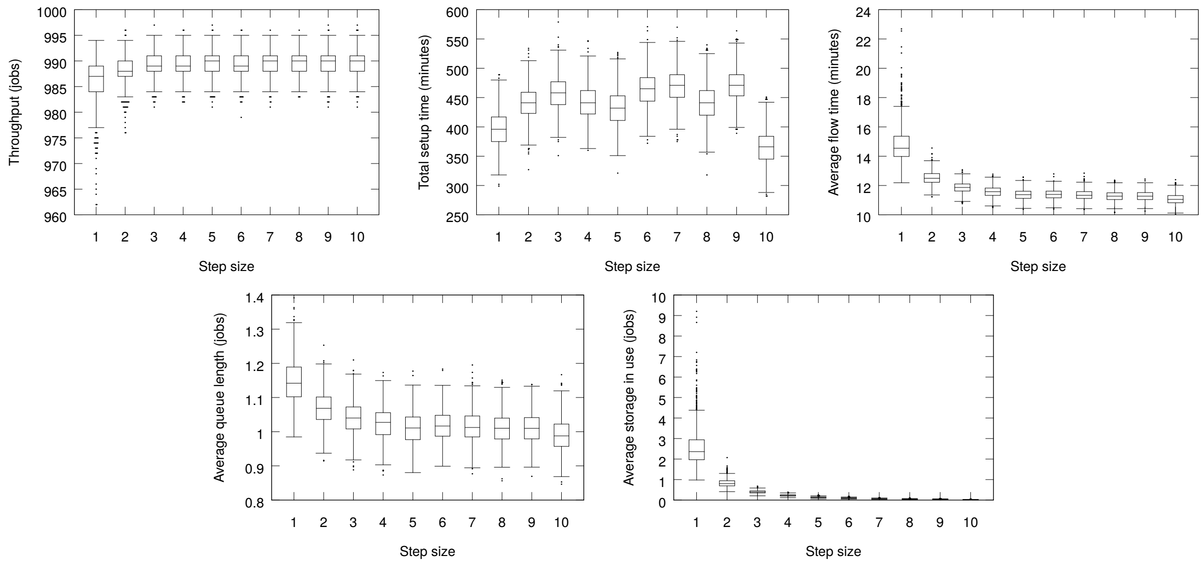

Figure 5: Results experiment 2: Non-uniformly distributed job types - FFW with different step sizes, 1000 repetitions.

Although the introduction of the foraging for work model [27] was followed by a back and forth of critiques [8, 9, 21, 22, 28] it is here shown to be an effective abstraction, even if it does not fully explain insect behaviour in nature.

More division of labour models have been proposed that have not been considered in this work, such as [1, 2, 6, 10, 20]. The works by [2, 20] are versions of the fixed threshold model which are not competitive with the reinforced threshold model [25] and its improved versions. The evolutionary threshold [6], network [1], and social inhibition [10] models are other alternatives that have not been used before and should be studied in future work.

Since FFW and ATA seem to excel in different areas, it may be worthwhile to investigate possible hybrid strategies in order to get the best of both worlds. This could result in an approach that is better than either individual algorithm.

REFERENCES

[1] Samuel N. Beshers and Jennifer H. Fewell. 2001. Models of Division of Labor in Social Insects. *Annual Review of Entomology* 46 (Jan. 2001), 413–440. https://doi.org/10.1146/annurev.ento.46.1.413

[2] Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. 1996. Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies. *Proceedings of the Royal Society of London B* 263, 1376 (Nov. 1996), 1565–1569. https://doi.org/10.1098/rspb.1996.0229

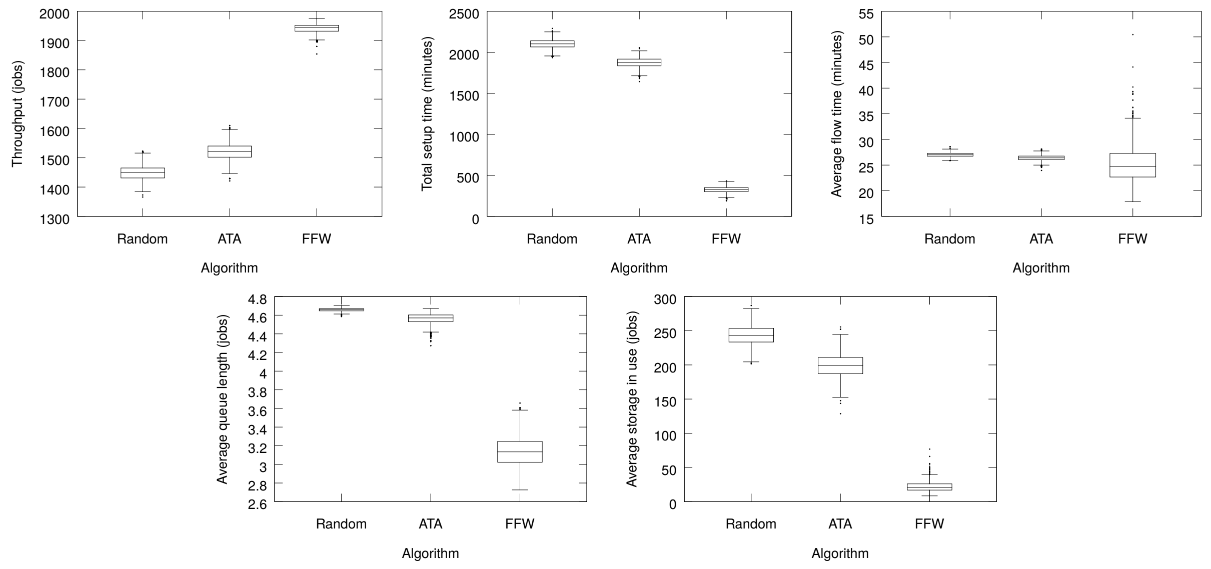

Figure 6: Results experiment 3: Non-uniformly distributed job types and heavy load (twice as many jobs), 1000 repetitions.

[3] Mike Campos, Eric Bonabeau, Guy Théraulaz, and Jean-Louis Deneubourg. 2000. Dynamic Scheduling and Division of Labor in Social Insects. *Adaptive Behavior* 8, 2 (March 2000), 83–96. https://doi.org/10.1177/105971230000800201

[4] Eduardo Castello, Tomoyuki Yamamoto, Fabio Dalla Libera, Wenguo Liu, Alan F. T. Winfield, Yutaka Nakamura, and Hiroshi Ishiguro. 2016. Adaptive foraging for simulated and real robotic swarms: the dynamical response threshold approach. *Swarm Intelligence* 10, 1 (March 2016), 1–31. https://doi.org/10.1007/s11721-015-0117-7

[5] Vincent A. Cicirello and Stephen F. Smith. 2004. Wasp-like Agents for Distributed Factory Coordination. *Autonomous Agents and Multi-Agent Systems* 8, 3 (May 2004), 237–266. https://doi.org/10.1023/B:AGNT.0000018807.12771.60

[6] Ana Duarte, Ido Pen, Laurent Keller, and Franz J. Weissing. 2012. Evolution of self-organized division of labor in a response threshold model. *Behavioral Ecology and Sociobiology* 66, 6 (June 2012), 947–957. https://doi.org/10.1007/s00265-012-1343-2

[7] Ana Duarte, Franz J. Weissing, Ido Pen, and Laurent Keller. 2011. An Evolutionary Perspective on Self-Organized Division of Labor in Social Insects. *Annual Review of Ecology, Evolution, and Systematics* 42 (Dec. 2011), 91–110. https://doi.org/10.1146/annurev-ecolsys-102710-145017

[8] N.R. Franks, C. Tofts, and A.B. Sendova-Franks. 1997. Studies of the division of labour: neither physics nor stamp collecting. *Animal Behaviour* 53, 1 (jan 1997), 219–224. https://doi.org/10.1006/anbe.1996.9998

[9] Nigel R. Franks and Chris Tofts. 1994. Foraging for work: how tasks allocate workers. *Animal Behaviour* 48, 2 (aug 1994), 470–472. https://doi.org/10.1006/anbe.1994.1261

[10] Deborah M. Gordon, Brian C. Goodwin, and L.E.H. Trainor. 1992. A Parallel Distributed Model of the Behaviour of Ant Colonies. *Journal of Theoretical Biology* 156, 3 (June 1992), 293–307. https://doi.org/10.1016/S0022-5193(05)80677-0

[11] Anshul Kanakia, Behrouz Touri, and Nikolaus Correll. 2016. Modeling multi-robot task allocation with limited information as global game. *Swarm Intelligence* 10, 2 (June 2016), 147–160. https://doi.org/10.1007/s11721-016-0123-4

[12] Oran Kittithreerapronchai and Carl Anderson. 2003. Do ants paint trucks better than chickens? Markets versus response thresholds for distributed dynamic scheduling. In *Evolutionary Computation, 2003. CEC ’03. The 2003 Congress on.* IEEE, 1431–1439. https://doi.org/10.1109/CEC.2003.1299839

[13] Lakshmanan Meyyappan, Can Saygin, and Cihan H. Dagli. 2007. Real-time routing in flexible flow shops: a self-adaptive swarm-based control model. *Swarm Intelligence* 45, 21 (Nov. 2007), 5157–5172. https://doi.org/10.1080/00207540600871277

[14] Dick Morley. 1996. Painting trucks at general motors: The effectiveness of a complexity-based approach. In *Embracing Complexity: Exploring the application of complex adaptive systems to business.* Ernst & Young Center for Business Innovation, Cambridge, MA, 53–58.

[15] Dick Morley and Gregg Ekberg. 1998. Cases in Chaos: Complexity-Based Approaches to Manufacturing. In *Embracing Complexity: A Colloquium on the Application of Complex Adaptive Systems to Business.* Ernst & Young Center for Business Innovation, Cambridge, MA, 97–102.

[16] Richard E. Morley and C. Schelberg. 1993. An analysis of a plant-specific dynamic scheduler. In *Proceedings of the NSF Workshop on Intelligent, Dynamic Scheduling for Manufacturing Systems.* 115–122.

[17] Shervin Nouyan. 2002. Agent-Based Approach to Dynamic Task Allocation. In *Ant Algorithms. Third International Workshop, ANTS 2002, Proceedings*, Marco Dorigo, Gianni Di Caro, and Michael Sampels (Eds.). Springer, Berlin, Heidelberg, 28–39. https://doi.org/10.1007/3-540-45724-0_3

[18] Shervin Nouyan, Roberto Ghizzioli, Mauro Birattari, and Marco Dorigo. 2005. An Insect-Based Algorithm for the Dynamic Task Allocation Problem. *Künstliche Intelligenz* 19, 4 (2005), 25–31.

[19] Djamila Ouelhadj and Sanja Petrovic. 2009. A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling* 12 (aug 2009), 417–431. https://doi.org/10.1007/s10951-008-0090-8

[20] R.C. Plowright and C.M.S. Plowright. 1988. Elitism in social insects: a positive feedback model. In *Interindividual behavioral variability in social insects.* Westview Press, Boulder, CO, 419–431.

[21] Gene E. Robinson, Robert E. Page Jr., and Z.-Y. Huang. 1994. Temporal polyethism in social insects is a developmental process. *Animal Behaviour* 48, 2 (aug 1994), 467–569. https://doi.org/10.1006/anbe.1994.1260

[22] Simon K. Robson and Samuel N. Beshers. 1997. Division of labour and âĂŸforaging for workâĂŹ: simulating reality versus the reality of simulations. *Animal Behaviour* 53, 1 (jan 1997), 214–218. https://doi.org/10.1006/anbe.1996.0290

[23] Stephen F. Smith. 2005. Is Scheduling a Solved Problem? In *Multidisciplinary Scheduling: Theory and Applications*, G. Kendall, E.K. Burke, S. Petrovic, and M. Gendreau (Eds.). Springer, Boston, MA, 3–17. https://doi.org/10.1007/0-387-27744-7_1

[24] Guy Theraulaz, Eric Bonabeau, and Jean-Louis Deneubourg. 1995. Self-organization of Hierarchies in Animal Societies: The Case of the Primitively Eusocial Wasp Polistes dominulus Christ. *Journal of Theoretical Biology* 174, 3 (June 1995), 313–323. https://doi.org/10.1006/jtbi.1995.0101

[25] Guy Theraulaz, Eric Bonabeau, and Jean-Louis Deneubourg. 1998. Response threshold reinforcement and division of labour in insect societies. *Proceedings of the Royal Society of London B* 265, 1393 (Feb. 1998), 327–332. https://doi.org/10.1098/rspb.1998.0299

[26] Guy Theraulaz, Simon Goss, Jacques Gervet, and Jean-Louis Deneubourg. 1991. Task differentiation in Polistes wasp colonies: a model for self-organizing groups of robots. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats.* MIT Press, Cambridge, MA, USA, 346–355.

[27] Chris Tofts. 1993. Algorithms for Task Allocation in Ants. (A Study of Temporal Polyethism: Theory). *Bulletin of Mathematical Biology* 55, 5 (Sept. 1993), 891–918. https://doi.org/10.1016/S0092-8240(05)80195-8

[28] James F.A. Traniello and Rebeca B. Rosengaus. 1997. Ecology, evolution and division of labour in social insects. *Animal Behaviour* 53, 1 (Jan. 1997), 209–213. https://doi.org/10.1006/anbe.1996.0289