A PSO-based Approach to IoT Workflow Scheduling in Cloud and Edge

Logan Blokland

Abstract—**IoT devices are becoming more and more common every day and play a huge part of our everyday lives and with these devices often having low processing power and availability to resources they also make use of edge and cloud servers. With all this information needed to be processed from different devices, it must be decided on what and when they are processed. From this, the IoT workflow scheduling problem is where we must organize through the workflows that are generated from various IoT devices and decide how they should be executed. To solve this problem this project proposes a Particle Swarm Optimization (PSO) based algorithm that can find a near-optimal solution to IoT workflow scheduling in cloud and edge. This solution is developed to focus on reducing both the energy and the makespan of executing a set of workflows. With this proposed solution we thoroughly evaluate and compare its performance showing its advantages over standard techniques, alongside how it compares the current state-of-the-art techniques using benchmark workflows which show the algorithm's ability to efficiently find a near optimal solution.**

I. Introduction

The IoT workflow scheduling problem in cloud and edge is where we must organize through the workflows that are generated from various IoT devices and decide how they should be scheduled in available resources in cloud and edge. There are multiple options to execute the workflow, these being locally on the IoT devices, they could also be run on edge servers or they can be processed externally through cloud servers. This can be a complex problem as there are many factors that need to be taken into account for optimizing different objectives such as energy consumption, response time, or other aspects on a large scale.

IoT workflow scheduling is an important problem with the increasingly large amount of IoT devices producing information that needs to be processed optimally. As many devices have limited processing power, memory or even limited power consumption, having effective workflow scheduling can optimise these resources including cost making it an important aspect study. To process all of this information the applications can rely on cloud servers, edge servers or even the devices themselves for computing resources. This workflow scheduling problem is a combinatorial problem as we aim to find the optimal assignment of tasks into a set of resources in the optimal order which in turn has a large number of possible combinations. There are current solutions available, but the aim of this project is to develop an algorithm that can effectively find near-optimal solution of distribution plans to optimally process the incoming data in real-time from outputting IoT devices.

This project was supervised by Hui Ma, and Aaron Chen.

The proposed algorithm will be a swarm intelligence algorithm which are well known to handle combinatorial problems efficiently by being able to explore the large set of solutions quickly and converge on an optimal solution. The aim of this project is to propose and evaluate an PSO-based algorithm that can find a near optimal solution to IoT workflow scheduling in cloud and edge.

A way this project could affect the environment is through potentially minimising the energy consumption used by running these IoT devices, this likely will not have any major impact but may improve energy consumption slightly even if it does not affect the big picture. By optimising the scheduling using our objective function this project will have tangible effects on energy consumption by minimising the overhead introduced dy inefficient data processing. This consideration of energy efficiency is a step towards the goals of environmental sustainability as well as encouraging more sustainable technology usage. Apart from this energy consideration, this project not have any other major effects on the environment or sustainability.

II. Related work and background

The main methods used to solve scheduling problems are generally heuristic solutions [1], mathematical optimization approaches, or AI/Evolutionary approaches [2]. Heuristic approaches are commonly used due to them being relatively simple, efficient, and able to produce satisfactory results in a reasonable amount of time. These generally work by utilizing simple rule-based techniques to make decisions on how to allocate tasks. Heuristic solutions although they are generally easy to implement compared to other methods as well as being efficient, they are generally unable to handle more complex scenarios or may be unable to find globally optimal solutions. Evolutionary approaches instead generally provide more complete solutions with a more comprehensive exploration of the search space, and as such this project will be focused of evolutionary approaches. Swarm intelligence [3] is a key idea that contains PSO, swarm intelligence relies on many different beings of a swarm communicating with each other allowing them to quickly and effectively explore the search space, this allows them to work efficiently on combinatorial problems like task allocation [3].

A recent paper was published where a method is proposed in which a memetic genetic algorithm can be used to solve the workflow scheduling problem [4]. This technique proposes a representation in which the solution is represented by a chromosome. Within this chromosome, each gene corresponds to a unique task in a workflow. The value of each gene in this chromosome represents the device that each task will be assigned to by using the device's type and index number. This sequential list representation is similar to what is being used in the proposed PSO algorithm as it is a clear and efficient way of representing the task allocation. Instead in our solution rather than using a tuple to reference the devices, we will be using a single index pointing to a device. This method also utilized a local search algorithm which in turn reduced the communication time and the cost required for execution. The results of this paper are going to be used to compare the performance of the developed PSO algorithm, comparing the same outputs shown in the paper, such as makespan and energy consumption. Using this we will be able to verify if our algorithm is able to reach our goal of achieving near-optimal results and to see how it compares to the GA algorithms shown in the paper. While these GA solutions perform well at finding optimal solutions, without the aid of local search, GA generally requires a larger population size, as well as requiring more generations to converge than PSO would and being a slower algorithm computation-wise in general [5] [6]. This is important as with scheduling in real time PSOs ability to quickly converge on an optimal solution makes it a promising solution to this problem.

III. IoT Workflow Scheduling

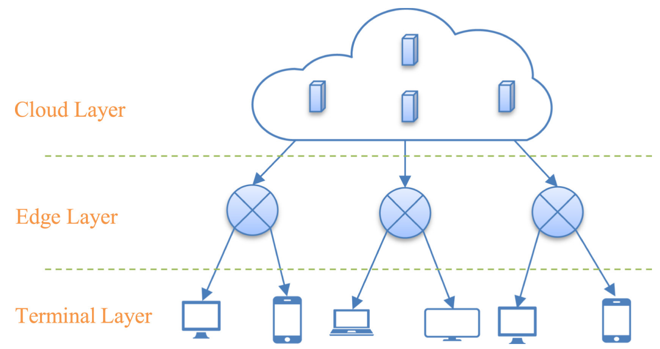

Fig. 1. Overview of scheduling problem

The workflow scheduling problem is where we must organize through the workflows that are generated from various IoT devices and decide how they should be executed. There are multiple options to execute the workflow, these being locally on the IoT devices, they could also be run on edge servers or they can be processed externally through cloud servers. This can be a complex problem as there are many factors that need to be taken into account for optimizing different objectives such as energy consumption, response time, or other aspects on a large scale. This workflow scheduling problem is a combinatorial problem as we aim to find the optimal assignment of tasks into a set of resources in the optimal order which in turn has a large number of possible combinations

Within this problem, we define a workflow WF which contains the IoT device that generated the workflow, and a directed acyclic graph that contains the information of each task t as a node and its dependencies represented by an edge on the graph (t, t'). Alongside this, we also have the generated solution Y which contains the information of allocation configurations according to each task in each given workflow to a device. These devices are chosen from a set of either IoT devices, fog servers, or cloud servers. Each of these devices can have varying attributes including the computing capacity, power consumption, and number of devices.

$$\mathcal{O}_{\text{energy}}(Y) = \sum_{i=1}^{K} \text{Energy}(WF_i, Y) \tag{1}$$

$$\mathcal{O}_{\text{makespan}}(Y) = \max_{i \in \{1,\ldots,K\}} CT(WF_i, Y) \tag{2}$$

$$\mathcal{O}_{\text{combined}}(Y) = w\tilde{\mathcal{O}}_{\text{makespan}}(Y) + (1-w)\tilde{\mathcal{O}}_{\text{energy}}(Y) \tag{3}$$

The fitness evaluation that needs to be minimised is calculated using the equation defined in 3. The objective function we are using allows for the balance of importance of either makespan or energy. This can be done by modifying the weight w, which is a value with 0 and 1 that will control the ratio of importance to consider between the two values. Through this we can analise the particles fitness in multiple different ways including makespan, energy consumption, or both. Both these energy and makespan calculations can be found in IotGA [4].

The energy calculated in 1 is done by finding the energy taken by each task and an in-depth equation can be found in [4]. But in a brief overview the energy is found by considering the energy consumption by any IoT device and the energy consumed from data communication between any device. So in the case of a task being executed on a device that is not the source IoT device it came from (e.g. could or edge devices), the energy will be calculated by considering both the idle power consumption of the IoT device and the power consumption from the data communication.

As for the makespan calculation, it can simply be found by going through each task in a workflow and finding the completion time of each task, then returning the max. This completion time must consider the communication delay between devices, as well as the dependencies that the current task has. These dependencies can be of two types, the first is a workflow dependency where one task depends on another in a given workflow. The other is a allocation dependency, where if both tasks are allocated to the same device as they must be executed sequentially.

IV. PSO-BASED IoT WORKFLOW SCHEDULING

A. Representation of solution

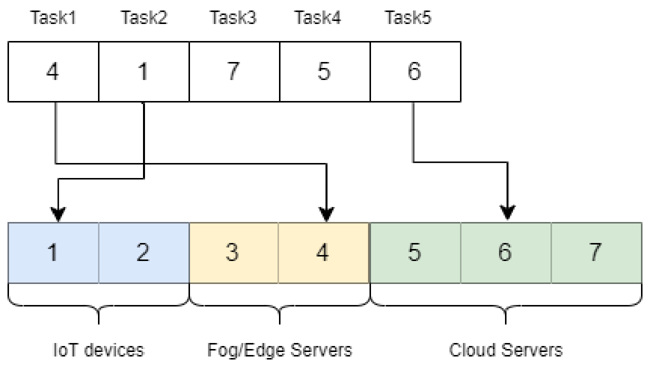

Fig. 2. Representation of PSO particle

Each solution is represented by a particle as shown in Fig 2. Permutation encoding is used to represent each particle as a list of numbers. Each index in the particle will correspond to a single task and will be ordered in the logical order that tasks are stored, e.g., index 1 = wf1 task 1, index 2= wf1 task 2, index 3 = wf2 task 1 etc...

The value that is stored in each of these task indexes will correspond to a unique device that the task will be assigned to. This is represented by a number that when rounded to the nearest integer can be used to index a unique device.

Each of the devices used will be assigned a unique number that is used to index them from the particles representation. The way in which the devices are ordered is something to consider, as the way in which are aligned may have an effect on how efficiently the algorithm may perform, as they likely can be grouped in some way that improves efficiency. In our solution the devices are being grouped in the order they were created, which are the three types of devices, first IoT devices second edge/fog servers then the cloud servers. This orders them in a way in that ranks them using processing speed so that the lowest speeds are first (e.g., IoT devices) then it moves to faster computation speed the further down the list you go (Fog/Edge to cloud servers). Although in our case it is not necessary to take this into account, but if various devices within the same group have varying speeds or capacities, it would be a good idea to logically arrange them within their group in an order that might be used to sort the groups, e.g., from slowest to fastest.

The population is created by randomly creating N particles. Each particle will be assigned a randomly selected number within the bounds of the number of devices onto each of the given tasks. For example a simple particle of (4, 2, 1, 6) would assign task 1 to device 4, task 2 two to device two, task 3 to device one, etc...

B. Fitness Evaluation

The fitness evaluation of each particle will be calculated using the equation 3. Through this we can analise the particle's fitness in multiple different ways including makespan, energy consumption, or both, depending on the weight set. For the majority of our testing and all of the results shown in this paper, a weight of 0.5 was set so that we could obtain an equal balance of both makespan and energy. But If need be, this can be changed so that either aspect could have priority over another.

V. EXPERIMENT EVALUATION

For the evaluation, I will be comparing this PSO method against other various methods by using the same datasets and running each method 30 times using the same random seeds for each of the runs. I will be comparing the results of each method's total makespan and other relevant values such as the convergence speed of each algorithm. When running these algorithms we need to run each one at least 30 times so that we can make sure the results produced are valid and of statistical significance. When running these tests we will be using the same set of random seeds (for each 30+ tests) on each of the algorithms that we will be comparing to make sure the results are fair.

On each of the problems that will be tested on, the results can be collected analising both the total makespan of the final schedule and the total Energy consumption. Along side this evaluation can also be done on each of the algorithms' convergence curves in order to see which of the options converge on an optimal solution the most efficiently.

A. Datasets

The main datasets that will be used to analyse the performance of the proposed algorithm are the problem instances that were used in the IoTGA paper [4].

The main source of data for evaluating the performance of the algorithm are the problem instances that were used in the paper [4]. These problem instances will be used as a benchmark for assessing the algorithm's ability to produce near-optimal solutions. The datasets used are that of real-world workflows, allowing for an accurate insight of how this algorithm would perform in real-world scenarios.

B. Development Tools and Methods

The main tool used in this project was the use of python, as this language has a lot of support for artificial intelligence techniques and algorithms and has many different statistic tools which were extremely useful during the evaluation process. In python the DEAP framework was used which provided key tools and structure streamlining the process of developing the PSO algorithm.

There was also access to Labs and computers through the university were used to aid the development of this project. An extremely useful tool that was used to aid in the evaluation of the performance of the solution was the ECS computing grid. This allowed for me to send out jobs to be run that can use superior processing power as well as being able to schedule multiple jobs at once. This makes the process of analysis much more efficient as I was able to schedule mainly 30 or any number or runs of an algorithm at once and allow the job to process overnight leaving the results to be compared and analysed without worrying about running the program on my machine.

C. Parameter Settings

The setting parameter settings used while evaluating the performance of the different algorithms it is important to keep all of the parameter setting the same as to be sure the results are fair to one another. As such, in order to compare results with IoTGA [4] the population size and maximum number of generations were adopted as 50 and 300 respectively. As for PSO-specific parameters such as the inertia weight, max velocity, and accelerate constants c1 and c2, these parameters are currently set to be as generally recommended values [7] such that $c1 = c2$ and are in a range of 1-3.

Parameter	Value
Number of IoT devices	4
Number of Fog/Edge servers	6
Number of Cloud servers	3
Bandwidth of LAN	2000 KB/s
Bandwidth of WAN	500 KB/s
Delay between IoT and Fog	0.5 ms
Delay between IoT and Cloud	30 ms
Computing capacity of IoT devices	500 MIPS
Computing capacity of fog servers	3000 MIPS
Computing capacity of cloud servers	4000 MIPS
Idle Power Consumption of IoT device	0.3 W
CPU Power Consumption of IoT device	0.9 W
Transmission Power of IoT devices	1.3 W

TABLE I
SYSTEM PARAMETERS

In table I above are the parameters specific to the IoT workflow scheduling problem. These parameters are defined in [8] and will be used across all competing algorithms and workflow problems. From [8] these parameters for things such as bandwidth, computing capacity and communication delay are commonly used settings that are used in both real-world applications and in other literature such as in [4].

D. Results

	Makespan	Energy	Combined fitness
GA	17.94 ± 3.03	11.53 ± 7.12	0.115 ± 0.029
IoTGA	15.80 ± 2.10	7.08 ± 3.19	0.092 ± 0.014
PSO	13.32 ± 4.50	8.32 ± 5.66	0.084 ± 0.027

TABLE II
COMBINED ENERGY AND MAKESPAN COMPARISON

Above we can see table II which shows the average total makespan, energy, and the combined weighted fitness of each algorithm being compared. After performing a Wilcoxon ranked test with a significance level of 0.05 on the obtained results, it shows that on the compared populations both IoTGA and PSO reject the null hypothesis when comparing its energy against standard GA. From this analysis we can conclude that there is statistical evidence to support that the distribution between the compared data has shifted away from zero to a significant degree validating its significance.

When performing a Wilcoxon ranked test on makespan and the fitness, it shows that it fails to reject the null hypothesis. We can conclude from this that the overall fitness and makespan PSO does not have statistical evidence to support that the distribution between has shifted away from zero to any significant degree.

Although through the wilcoxon ranked test no statistical difference can be found, we can still draw conclusions on the date based on their performance. Even though some did fail the wilcoxon test, their p-vales were close such as 0.07, alongside this we may need more sample data to fully conclude its significance, but overall with these results we do not have enough evidence to confidently assert a difference. Although as can be seen in the table, the worst performing algorithm across all aspects was standard GA, with both IoTGA and PSO being able to outperform it in obtaining lower fitness values.

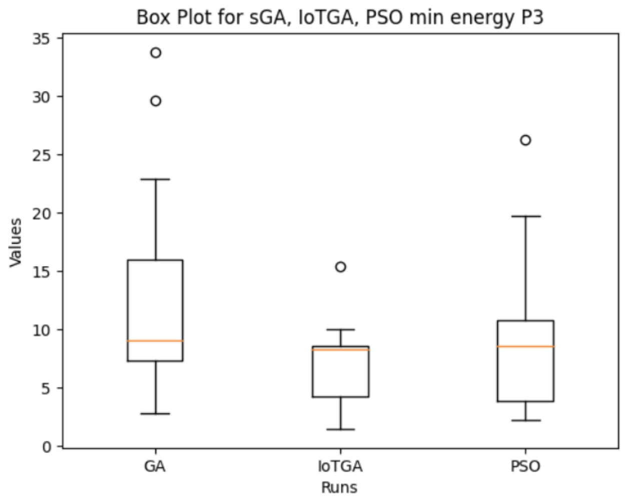

Fig. 3. Boxplot showing distribution of minimum energy obtained by competing algorythims

As seen in Table II the energy consumption of both GA and PSO have relatively high standard deviations when compared to IoTGA. We can explore the reason for this by using the box plot above in Figure 3. As seen clearly in the plot, the median value of all three algorithms are very similar but both PSO and GA have much higher upper whiskers and outliers. As IoTGA is much more efficient with its upper ranges, it has a much smaller standard deviation compared to the others. This is due to its local search that is implemented allowing for more rapid improvements of solutions. While this is true it can still be seen that PSO does outperform GA with is much lower inter-quartile range, on average providing better solutions than GA.

E. Further Analysis

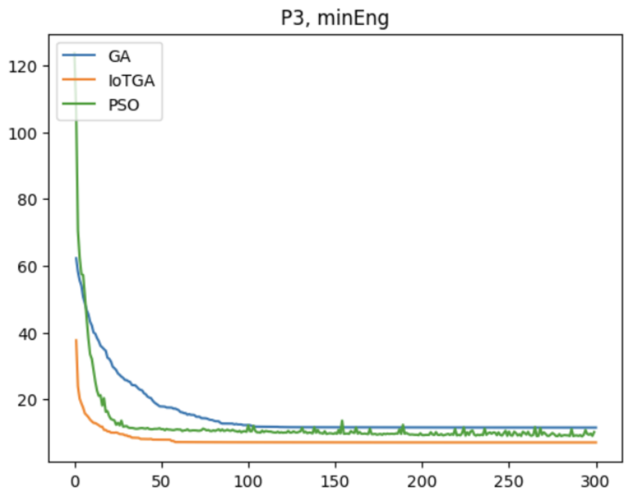

Fig. 4. Convergance of PSO

The convergence curves shown in the graphs above show the performance of the three competing algorithms averaged over 30 runs on a test set of workflow data.

After analyzing the minimum energy obtained in this graph 4 it is clear that the algorithm is able to quickly converge towards a minimum value showing that it can achieve a high-quality solution to the given problem dataset. It can be seen that again GA is the worst performing of the three algorythims, both in the case of its relatively slow convergence speed, as well as its final overall value. On the other hand our proposed algorithm is able outperform standard GA (sGA), as can be observed in the Fig 4, it is able to converge much faster as it reaches its final range of values around 50 generations, while standard GA needs 100 generations to converge to its final value. While it does perform well compared to sGA, it does not perform as well as IoTGA in both convergence and the final obtained value, this is due to IoTGAs utilization of local search techniques which allows for faster convergence and a better obtained solution.

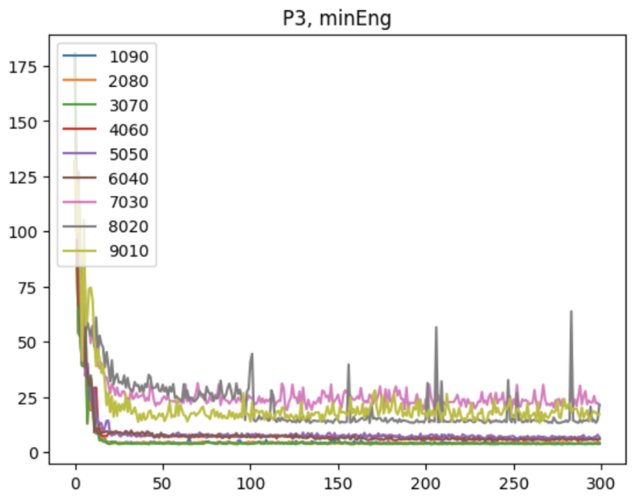

Fig. 5. Effect of different weights on energy

The above graph shows the different convergence curves of the total energy obtained by our PSO algorithm using a number of different weights for the computed fitness values. Each line on the graph represents a different ratio of makespan to energy, for example, 2080 represents a weight of 0.2 makespan and 0.8 energy. As seen in the graph once the ratio reaches 0.7:0.3 the results begin to become much more fluctuates, as the energy has less and less influence on the value of the fitness. And on the other end a ratio of 0.6:0.4, there is little improvement to be seen amongst the differing values. From these results we can conclude that it is optimal to use a value that roughly balances the influences of both characteristics in order for the optimal value to be found. We can also see that the 50:50 ratio that has been used in all of the main evaluations is an optimal way to run this algorithm.

VI. Conclusions

A. Future work

Some future work that could be done would be to utilise this representation of the problem to further improve on the results obtained, some ways could include to have a better particle initialization or introducing a local search technique to further improve on solutions.

One possible local search technique that could be implemented that we discussed was where you are to look at the longest edge on the graph or the task with the largest communication time. When this edge is found you can then assign either task to a random new group of devices or allocate both tasks onto the same device group in order to minimise communication time. A similar technique could be implemented when looking at the initialization of the particles as well such as putting an importance on communication time between tasks on different machines or the processing time of each task such as the technique implemented in [9].

B. Conclusion

In conclusion, this project tackles the key problem of IoT workflow scheduling, a challenging issue brought on by the large number of IoT devices in our daily lives. Effective workflow scheduling becomes necessary because these devices frequently have low processing power and few resources, dictating when and how certain tasks are done. As workflows can be done locally on IoT devices, remotely through cloud servers, or locally on edge servers, the complexity increases, needing careful consideration of issues like energy usage and makespan.

To address this complex issue, we put forth a Particle Swarm Optimization PSO-based method that finds near-optimal scheduling for IoT workflows in cloud and edge contexts. Our algorithm showed its ability in obtaining near-optimal solutions through performing comparisons against other approaches utilizing benchmark workflows. In conclusion, our algorithm presents a potential approach to solving the problems associated with IoT workflow scheduling and proposes points where further research can be performed looking into the effectiveness of PSO on the IoT workflow scheduling problem. The significance of effective and sustainable resource utilization cannot be emphasized, and this initiative is a step in that direction as we continue to see the extensive integration of IoT devices into our lives.

References

[1] J. Wang and D. Li, "Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing," *Sensors*, vol. 19, no. 5, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/5/1023

[2] B. M. Nguyen, H. Thi Thanh Binh, T. The Anh, and D. Bao Son, "Evolutionary algorithms to optimize task scheduling problem for the iot based bag-of-tasks application in cloud–fog computing environment," *Applied Sciences*, vol. 9, no. 9, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/9/1730

[3] J. Odili, M. N. M. Kahar, A. Noraziah, and S. F. Kamarulzaman, "A comparative evaluation of swarm intelligence techniques for solving combinatorial optimization problems," *International Journal of Advanced Robotic Systems*, vol. 14, no. 3, p. 1729881417705969, 2017.

[4] A. Saeed, G. Chen, H. Ma, and Q. Fu, "A memetic genetic algorithm for optimal iot workflow scheduling," in *Applications of Evolutionary Computation: 26th European Conference, EvoApplications 2023, Held as Part of EvoStar 2023, Brno, Czech Republic, April 12–14, 2023, Proceedings*. Springer, 2023, pp. 556–572.

[5] C. A. S. Lima Jr, C. M. F. Lapa, C. M. do NA Pereira, J. J. da Cunha, and A. C. M. Alvim, "Comparison of computational performance of ga and pso optimization techniques when designing similar systems–typical pwr core case," *Annals of Nuclear Energy*, vol. 38, no. 6, pp. 1339–1346, 2011.

[6] S. Panda and N. P. Padhy, "Comparison of particle swarm optimization and genetic algorithm for facts-based controller design," *Applied soft computing*, vol. 8, no. 4, pp. 1418–1427, 2008.

[7] Y. He, W. J. Ma, and J. P. Zhang, "The parameters selection of pso algorithm influencing on performance of fault diagnosis," in *MATEC Web of conferences*, vol. 63. EDP Sciences, 2016.

[8] M. Goudarzi, H. Wu, M. Palaniswami, and R. Buyya, "An application placement technique for concurrent iot applications in edge and fog computing environments," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1298–1311, 2020.

[9] B. Jarboui, S. Ibrahim, P. Siarry, and A. Rebai, "A combinatorial particle swarm optimisation for solving permutation flowshop problems," *Computers & Industrial Engineering*, vol. 54, no. 3, pp. 526–538, 2008.