MODIFIED AND HYBRID ARTIFICIAL FISH SWARM ALGORITHM FOR SOLVING DISCRETE FUZZY MULTI-OBJECTIVE UNRELATED PARALLEL MACHINE SCHEDULING PROBLEM

AZHAR MAHDI IBADI

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2026

Awang Had Salleh
Graduate School
of Arts And Sciences

Universiti Utara Malaysia

PERAKUAN KERJA TESIS / DISERTASI
(Certification of thesis / dissertation)

Kami, yang bertandatangan, memperakukan bahawa
(We, the undersigned, certify that)

AZHAR MAHDI IBADI

calon untuk Ijazah
(candidate for the degree of) PhD

telah mengemukakan tesis / disertasi yang bertajuk:
(has presented his/her thesis / dissertation of the following title):

"MODIFIED AND HYBRID ARTIFICIAL FISH SWARM ALGORITHM FOR SOLVING DISCRETE FUZZY MULTI- OBJECTIVE UNRELATED PARALLEL MACHINE SCHEDULING PROBLEM"

seperti yang tercatat di muka surat tajuk dan kulit tesis / disertasi.
(as it appears on the title page and front cover of the thesis / dissertation).

Bahawa tesis/disertasi tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan, sebagaimana yang ditunjukkan oleh calon dalam ujian lisan yang diadakan pada : **29 September 2025.**
That the said thesis/dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: ***29 September 2025.***

Pengerusi Viva *(Chairman for Viva)*	: **Assoc. Prof. Dr. Syariza Abdul Rahman**	Tandatangan *(Signature)*	
Pemeriksa Luar *(External Examiner)*	: **Assoc. Prof. Dr. Kavikumar Jacob**	Tandatangan *(Signature)*	
Pemeriksa Dalam *(Internal Examiner)*	: **Assoc. Prof. Dr. Nazihah Ahmad**	Tandatangan *(Signature)*	
Nama Penyelia *(Name of Supervisor)*	: **Assoc. Prof. Dr. Rosshairy Abdul Rahman**	Tandatangan *(Signature)*	

Tarikh:
(Date) ***29 September 2025***

Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor or, in her absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to:

Dean of Awang Had Salleh Graduate School of Arts and Sciences
UUMCollege of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok

Abstrak

Masalah Penjadualan Mesin Selari Tidak Berkaitan (UPMSP) ialah masalah pengoptimuman diskret yang banyak digunakan dalam membuat keputusan industri. Walau bagaimanapun, masalah ini sering melibatkan ketidakpastian akibat pengukuran yang tidak tepat dan data yang tidak lengkap. Tambahan pula, mengenal pasti penyelesaian optimum menggunakan algoritma tepat adalah mencabar, terutamanya apabila melibatkan pelbagai objektif. Oleh itu, kajian ini mencadangkan Algoritma Sekawan Ikan Buatan (AFSA) yang diubah suai, pada asalnya direka untuk masalah berterusan, bagi meminimumkan masa siap berbilang objektif kabur dan kelewatan keseluruhan. Algoritma yang diubah suai ini menggabungkan tingkah laku aspirasi, penambahbaikan parameter, serta kaedah transformasi untuk menyesuaikan dengan UPMSP diskret. Tiga set masalah dengan saiz kerja dan mesin yang berbeza dijana untuk menilai keberkesanan algoritma AFSA yang diubah suai. Ukuran statistik termasuk nilai minimum, maksimum, min, sisihan piawai, dan ujian pangkat bertanda Wilcoxon digunakan untuk membandingkan prestasi dengan AFSA standard dan lima variasi AFSA yang dicadangkan dalam kajian lepas. Algoritma yang diubah suai mencapai nilai minimum terbaik masa siap dan kelewatan keseluruhan dalam tiga daripada sepuluh data kecil dan dalam sembilan daripada sepuluh data sederhana dan besar. Ini menunjukkan prestasi algoritma yang dicadangkan jauh lebih baik secara signifikan daripada algoritma sedia ada, khususnya bagi masalah bersaiz sederhana dan besar. Walau bagaimanapun, disebabkan keputusan yang kurang memuaskan bagi saiz mesin empat hingga sepuluh, penambahbaikan dilakukan dengan menggabungkan AFSA yang diubah suai dengan algoritma carian kejiranan. Algoritma hibrid ini mendapat manfaat daripada penerokaan global AFSA dan eksploitasi tempatan yang kukuh melalui operasi kejiranan, menghasilkan keputusan yang lebih tepat. Perbandingan dengan tiga set algoritma termasuk AFSA yang diubah suai, metaheuristik tunggal, dan metaheuristik hibrid dijalankan untuk menilai prestasi. Hasil pengiraan mengesahkan bahawa algoritma hibrid mengatasi pesaingnya secara signifikan. Secara keseluruhan, kajian ini menyumbang kepada peningkatan kecekapan penjadualan kerja dan keberkesanan penyelesaian masalah diskrit terutamanya UPMSP menggunakan algoritma AFSA yang diubah suai dan hibrid, sekali gus meningkatkan kualiti penjadualan dalam pelbagai industri. Algoritma yang dicadangkan juga boleh diperluas untuk menyelesaikan gabungan pelbagai objektif dalam persekitaran penjadualan yang lebih kompleks seperti penjadualan tubuhan kerja atau penjadualan tubuhan aliran.

Kata Kunci: Algoritma Sekawan Ikan Buatan, Berbilang Objektif Kabur, Hibrid Metaheuristik, Algoritma Carian Kejiranan, Masalah Penjadualan Mesin Selari Tidak Berkaitan.

Abstract

The Unrelated Parallel Machines Scheduling Problem (UPMSP) is a discrete optimization problem widely applied in industrial decision-making. However, it often involves fuzzy uncertainty due to imprecise measurements and incomplete data. Moreover, identifying optimal solutions using exact algorithms is challenging, especially when multiple objectives are involved. Thus, this research proposes a modified Artificial Fish Swarm Algorithm (AFSA), originally designed for continuous problems, to minimize fuzzy multi-objective makespan and total tardiness. The proposed modified algorithm incorporates aspiration behavior, improved parameters, and a transformation method to adapt to discrete UPMSP. Three problem sets with varying job and machine sizes were generated to evaluate the modified AFSA algorithm. Using statistical measures including minimum, maximum, mean, standard deviation, and the Wilcoxon signed-rank test, its performance was compared with the standard AFSA and five of its variants proposed in the literature. The modified algorithm achieved the best minimum value of makespan and total tardiness in three of ten small instances and in nine of ten medium and large instances. This demonstrated that the proposed algorithm significantly outperformed the others existing algorithms, particularly for medium and large-sized problems. However, due to inferior results for machine sizes four to ten, improvements were made by hybridizing the modified AFSA with a neighborhood search algorithm. The hybrid algorithm benefits from AFSA's global exploration, and the robust local exploitation provided by neighborhood operations, leading to more superior results. A comparison with three sets of algorithms including the modified AFSA, single metaheuristics, and hybrid metaheuristics was conducted to assess performance. The computational results confirmed that the hybrid algorithm significantly surpassed its competitors. Overall, this study contributes to improving job scheduling and enhances the efficiency of solving discrete problems, particularly UPMSP using modified and hybrid AFSA, ultimately leading to better scheduling quality across various industries. The proposed algorithms can also be extended to solve combinations of multi-objectives in more complex scheduling environments, such as job shop or flow shop scheduling.

Keywords: Artificial Fish Swarm Algorithm, Fuzzy Multi-Objective, Hybrid Metaheuristics, Neighborhood Search Algorithm, Unrelated Parallel Machine Scheduling Problem

Acknowledgement

"In the name of Allah, the Most Gracious and the Most Merciful"

Alhamdulillah, it is impossible for me to accomplish this thesis without the blessings from Allah. I am deeply grateful for His guidance and support throughout this journey. My greatest thanks go out to Dr. Rosshairy Abd Rahman, my wonderful supervisor, for all the support, encouragement, and insightful criticism she has given me. This research would not have been possible without her knowledge and advice. Working under her supervision will become one of the most unforgettable experiences in my life.

To my family, thank you for your unwavering love and encouragement. Your belief in me has been a constant source of motivation.

Special thanks to my colleagues and friends, whom I met in the first days at UUM, for their camaraderie and for creating a stimulating and collaborative environment. Your support and friendship have made this journey more enjoyable.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ABC	Artificial Bee Colonies
ACO	Ant Colony Optimization
AFSA	Artificial Fish Swarm Algorithm
B&B	Branch and Bound
DRs	Dispatching Rules
EAs	Evolutionary Algorithms
FA	Firefly Algorithm
FOA	Fruit Fly Optimization Algorithm
FSSP	Flow shop scheduling problem
GA	Genetic Algorithm
GWO	Grey Wolf Optimizer
HABC	Hybrid Artificial Bee Colony Algorithm
JSSP	Job Shop Scheduling Problem
MILP	Mixed Integer Linear Programming
MP	Mathematical Programming
MSPs	Machine Scheduling Problems
NP-Hard	Nondeterministic Polynomial Time
PMSP	Parallel Machine Scheduling Problem
PSO	Particle Swarm Optimization
SA	Simulated Annealing
SIAs	Swarm Intelligence Algorithms
SSA	Salp Swarm Algorithm
SSAFA	Enhanced Salp Swarm Algorithm Based on Firefly Algorithm
SSO	Simplified Swarm Optimization
TFNs	Triangular Fuzzy Numbers
TrFNs	Trapezoidal Fuzzy Numbers
TS	Tabu Search
UPMSP	Unrelated Parallel Machines Scheduling Problem
WOA	Whale Optimization Algorithm
WOAFA	Whale Optimization Algorithm and Firefly Algorithm

CHAPTER ONE

INTRODUCTION

1.1 Introduction

Scheduling is the process of organizing, planning, and assigning tasks to achieve specific goals or objectives within given constraints. It is widely used in various domains to guarantee effective use of tasks, adherence to deadlines, and the smooth execution of operations, directly impacting people's daily lives by influencing decision-making in numerous areas. Some examples include industrial operations, production management, manufacturing systems, healthcare (involving patients and hospital equipment), educational institutions (classes and educators), logistical structures (vessels and shipyards), computer programs, and even city-traveling salesmen. Meanwhile, Machine Scheduling Problems (MSPs) involve the planning and organizing of jobs that must be performed on machines in a predetermined order. This process aims to maximize the efficient use of available resources. Thus, improving machine scheduling methods is crucial for increasing productivity, reducing input costs, and enhancing overall quality of life.

1.2 Background of the Machine Scheduling Problem

As the world evolves toward higher efficiency, appropriate executions of machine scheduling strategies are required to fulfill these expanding expectations. MSPs are a key topic in operations research. It is an extremely active decision-making type with significant practical importance in the industrial and service industries. Notably, practical machine scheduling issues come in various forms. They occur across various industries,

including production scheduling, processing scheduling, crew scheduling, and airport gate scheduling. Building on this, the scheduled construction process is crucial since it can significantly affect a company's productivity (Fonseca-Reyna et al., 2018).

Graham et al. (1979) introduced three field notations $\alpha|\beta|\gamma$ known as a triplet, for identifying a basic scheduling problem. The information provided by the triplet is employed to detail a scheduling problem. Furthermore, the α field defines the machine environment, information about processing characteristics can be discovered in the field β, and γ denotes the scheduling problem's objective function under consideration (Ali, 2020). Based on the machine's environment, problems with scheduling can be classified into five major categories, illustrated in Figure 1.1 as follows:

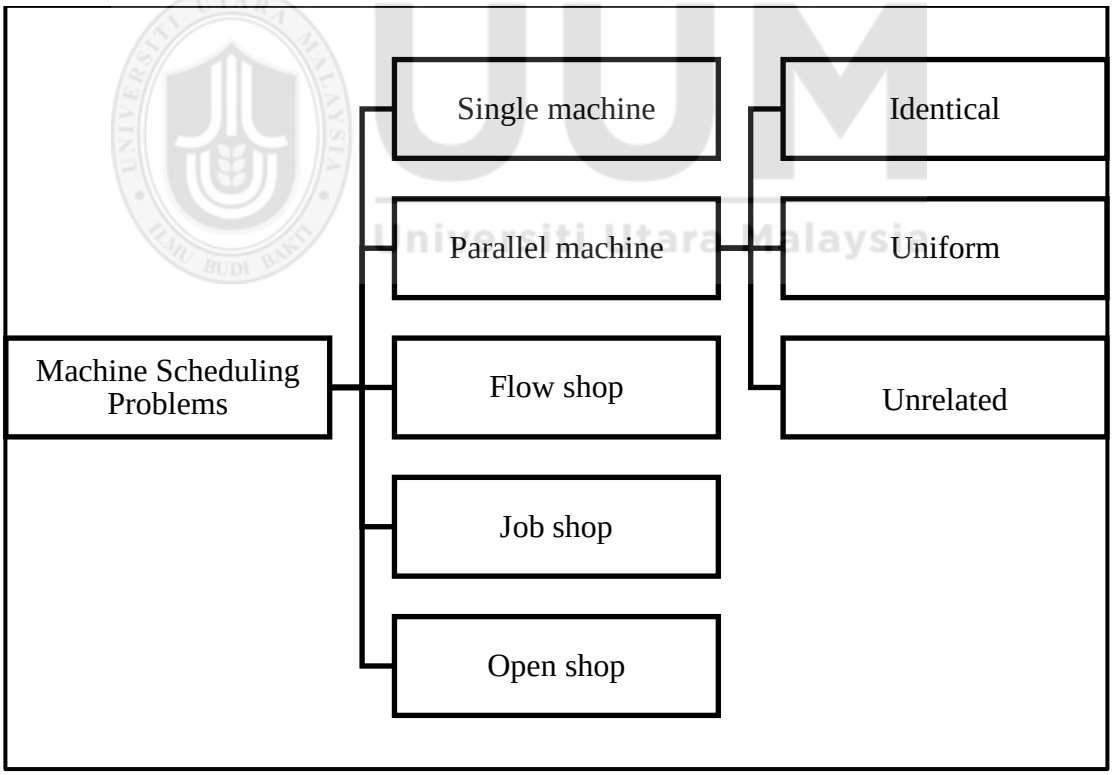


Figure 1.1. A classification of scheduling problems (Behnamian, 2016)

In models for a single machine, only one machine is involved, and the procedures have one operation conducted by this machine (Sabti, 2017). This model is the most straightforward of all environments and supports understand other scheduling problems. However, in parallel machine scheduling, *N* jobs and *M* machines are involved, and all jobs can be operated on any one of the existing machines. There are three distinct kinds of parallel machines:

- Identical (P): Every machine takes the same processing time for a job.
- Uniform related (Q): Machine speed affects a job's processing time. Machines that are parallel and uniformly related operate at different speeds. The relationship $p_{ij} = p_j/s_i$, where s_i stands for the speed of the machine i, can be used to determine how long it will take to process work in relation to machine speed.
- Unrelated (R): Consists of unrelated machines operating in parallel with different features, resulting in a time difference in job-dependent processing speeds p_{ij}.

It is important to acknowledge that parallel machines are not necessarily identical. Accordingly, unrelated parallel machines are machines that are not identical to one another. In particular, an Unrelated Parallel Machine Scheduling Problem (UPMSP) seeks to allocate jobs to unrelated machines that can process them in parallel without interfering with one another, this suggests that there is no connection between the machines.

In operations research, the Parallel Machine Scheduling Problem (PMSP) is one of the most complex challenges (Salimifard et al., 2019), requiring decisions on both machine assignment and job sequencing. This problem proves difficult to solve and is classified as Nondeterministic Polynomial Time (NP-hard) (Garey & Johnson, 1979). Hence, to

effectively address the PMSP, it is crucial to simultaneously define both the allocation and ordering strategies for the given parallel machines and jobs (Ezugwu et al., 2018). Thus, Pinedo (2008) highlighted that parallel machines provide a theoretically and practically significant environment, and a bank of parallel machines is theoretically a flexible flow shop special case and a single machine generalization. A bank of parallel machines is significant from a practical standpoint since resources are highly likely to exist in parallel. In addition, the significance PMSP stems from their capacity in optimizing the utilization of various machines and assigning particular jobs to fulfill the needs of the consumer. This is critical for industries where optimizing resources is required, such as maximizing productivity and minimizing machine overall time. Furthermore, PMSP models can be adapted for various scenarios, ranging from small-scale operations to large, complicated systems. This makes them a valuable tool for an extensive variety of industries, particularly in high-demand environments like data centers or manufacturing facilities (Salimifard et al., 2019). The PMSP difficulties stem from its complexity, which has getting more common, leading to its classification as NP-hard (Salimifard et al., 2019; Gray & Johnson, 1979). This suggests that finding a best solution is computationally costly and becomes more challenging as the number of jobs and machines increases.

The environment is referred to as a flow shop, job shop, and open shop, where there are N jobs and M machines, and each job has several operations O_j for processing the same set of jobs in a flow. If the number of $O_j = M$, this environment is known as a flow shop. Otherwise, the environment is known as a job shop or open shop. Accordingly, the problems of the flow shop involve a strict sequence of all operations to be conducted on

all jobs. In particular, each job in a job shop environment has its own sequence to be executed on machines and can adhere to a distinct path compared to other jobs. However, the scheduling problem with open shops is that each job's operations can occur in any sequence.

Based on the scheduling procedure, the scheduling problem was divided into single and multiple operation problems. Problems with single and parallel machines are subcategories of single-operation problems. Conversely, problems with flow shops, job shops, and open shops are subcategories of multiple operation problems (Lin & Huang, 2021).

1.3 Why Unrelated Parallel Machine Scheduling Problem?

MSPs is a crucial factor of resource management since competition is a prevalent activity in manufacturing environments. It focuses on allocating limited resources (such as time, machines, or budget) among different jobs to achieve the best possible outcome. This may involve optimizing a single objective, such as minimizing cost or time, or balancing multiple objectives, such as reducing delays while maximizing efficiency (Çolak & Keskin, 2022). Hence, MSPs focuses on determining which facility resources will perform which jobs, in what sequence, and at what time (Pınarbaşı, 2019). The PMSP has attracted ongoing interest in an extensive collection of production scheduling researchers and experts from engineering and scientific fields (Ezugwu & Akutsah, 2018). This heightened interest can be credited to the numerous parallel machine scheduling applications in contemporary industrial systems. Typical applications of PMSP include flexible manufacturing systems (Dang et al., 2021) and plastic molding industries (Salimifard et al., 2019), to name a few.

In addition, PMSP is an expansion of a single-machine problem, representing a real scenario in manufacturing. Furthermore, the incorporation of new restrictions to this problem leads to the development of additional PMSP variants, such as the UPMSP (Ezugwu & Akutsah, 2018). The UPMSP was defined by Đurasević and Jakobovi (2021) as a model of a problem with multiple components. Such components contain of N jobs that must be placed on M machines with differing capacities. Due to this, the appointed jobs are always performed at different rates. The UPMSP problem has wide industrial applicability, which is critical in manufacturing and production scheduling, such as in the textile industry. Improving solution techniques directly leads to significant real-world increases in efficiency, cost savings, and productivity.

Formally, the UPMSP can be expressed mathematically as a Mixed Integer Programming (MIP) Model (Ulaga et al., 2022; Unlu & Mason, 2010). The mathematical formulation employs the following notations for sets, parameters, and variables as follows:

N refers to the set of jobs, indexed $j = 1, \dots, N$,

M refers to the set of machines, indexed $i = 1, \dots, M$,

τ refers to the set of periods of time, indexed $t = 1, \dots, l$,

l is an upper bound of the last job's completion time on a machine (i.e., makespan) and is defined as $l \geq \max_i \sum_j \frac{p_j}{v_{ij}}$,

v_i refers to the speed of machine i; in the case of job-dependent machine speed (R_m), v_{ij} signifies the speed of machine i for job j,

p_{ij} symbolizes to the processing time of job j on machine i, defined as $p_{ij} = \frac{p_j}{v_{ij}}$,

${X^t}_{ij}$ is a binary variable, ${X^t}_{ij} = 1$, if job $j \in N$ start processing on machine $i \in M$ at time $t \in \tau$, ${X^t}_{ij} = 0$ otherwise.

Model UPMSP:

$$\sum_{i \in M} \sum_{t=0}^{l-1} {X^t}_{ij} = 1 \qquad j \in N, \tag{1.1}$$

$$\sum_{j \in N} \sum_{h=max(0,t-p_{ij})}^{t-1} {X^h}_{ij} \leq 1 \quad i \in M, t = 1, \dots, l, \tag{1.2}$$

$$C_{ij} \geq \sum_{i \in M} \sum_{t=0}^{l-1} (t + p_{ij}) {X^t}_{ij} \quad j \in N. \tag{1.3}$$

Constraint (1.1) assurances that every job begins processing on only one machine at a single point in time. Meanwhile, constraint (1.2) means allowing only one job to be processed at any given time on any machine. Constraint (1.3) signifies the completion time C_{ij} of job j. Overall, this UPMSP model serves as a foundational framework that is flexible enough to accommodate various objective functions and operational constraints. The UPMSP being studied is a generalization of the problem of identical and uniform parallel machines. UPMSPs are widely employed and have significant practical importance and relevance in many ways in the real world in both production scheduling and manufacturing systems. This includes a factory that manufactures customized heating, ventilation, and air conditioning equipment (Chi et al., 2022), semiconductor manufacturing (Chen et al., 2022), electronic assembly (Lin & Huang, 2021), and manufacturing electricity costs (Abikarram et al., 2019). In fact, it is also utilized in production scheduling operations at a television manufacturer (Ekici et al., 2019) and

many others. Since the UPMSP is the general type of other PMSPs, it is more realistic since the variability in machine speeds, technology, and types within the production environment. Therefore, processing times in various machines running in parallel are likely to differ for the same job.

The UPMSP is a critical discrete optimization problem widely used for decision-making in both industrial and non-industrial settings (Pulansari & Triyono, 2021; Agárdi & Nehéz, 2021). Scheduling performance is strongly influenced by Multi-Objective Optimization (MOO), and nearly half of the publications in this field address multi-objective formulations (Fu et al., 2021). Among these, makespan and total tardiness are the most frequently examined objectives, as they represent the most significant scheduling challenges across various application domains (Guevara-Guevara et al., 2022). Therefore, a study by Kongsri and Buddhakulsomsiri (2020) proposed a Mixed Integer Linear Programming (MILP) model for UPMSP with sequence-dependent setup time constraints that minimize makespan and total tardiness in a deterministic environment. However, sequence-dependent setup time constraints require a lengthy solution time, and exact methods such as linear programming models, which are used to solve this model, lack flexibility when many constraints and experiments are required to achieve appropriate solutions in a reasonable time frame. Furthermore, UPMSP will result in significant memory requirements and solution times due to the large number of jobs and machines and varying processing times for each job. This was confirmed by computational tests to determine optimal solution times, which are 83.59 hours for solving a small problem instance of only three unrelated parallel machines processing ten jobs.

1.4 Challenges of Unrelated Parallel Machine Scheduling Problems

The deterministic optimization paradigm is widely used in UPMSP, assuming that all parameters are known with certainty. Although this simplifies analysis, the assumption is often unrealistic in practice (Liao & Su, 2017). Many studies treat due dates, release dates, and processing times as fixed values; however, in real-world UPMSP environments, these parameters are frequently subject to uncertainty.

MSPs are highly concerned with uncertainty as it can lead to inefficiency and disruptions in planning. As a result, the uncertainty of UPMSPs has gained great interest in the operations research community in recent times in the literature. In addition, multi-objectives in UPMSPs could be made more complex by imprecise values, especially due to the ambiguity or imprecision of time estimations. Another challenge with UPMSPs is their difficulty level, which is categorized as NP-hard computationally. There isn't an algorithm that can currently generate an optimal solution in a time that can be expressed as a finite polynomial of the problem's dimension (Lee et al., 2002). Moreover, real-world scenarios frequently include dynamic changes and uncertainties, such as machine breakdowns or different job durations. Correspondingly, machine constraints and multiple objectives must be considered, and balancing them can complicate the scheduling process. Accordingly, different scheduling applications deal with uncertainty, such as in the process of scheduling patient appointments, multi-objective multi-product pipeline scheduling and multi-constraint weekly meals scheduling.

There are primarily three main approaches has been described in previous studies (Li & Ierapetritou, 2008; Gharehgozli et al., 2009; Behnamian, 2016; Rezaeian et al., 2020), to overcome uncertainty issues in the environment of MSPs. The first class is a stochastic scheduling problem, which is based on probabilistic theory and possibility theory. It

employs stochastic variables to represent processing conditions and parameters. Meanwhile, the second is a fuzzy scheduling problem according to fuzzy set theory, which models the processing parameters and conditions using a fuzzy number, and robust scheduling is the third one. In particular, robust strategies aim to create solutions capable of absorbing some unexpected events without rescheduling (Behnamian, 2016).

Stochastic probabilistic theory is a logical choice, yet it supposes that we comprehend the nature of uncertainty. It is explained in terms of events and their recurrence, which is represented by an arbitrary variable with a specific distribution of probabilities. However, the need for an extensive understanding of the statistical distribution of the unknown parameter is one of the main drawbacks and challenges of using probabilistic theory. Then, the fuzzy scheduling problem is a more effective way to model uncertainty and imprecision than the latter, particularly if no historical data exists (Behnamian, 2016). Meanwhile, the robust scheduling is assumed that the interval (or range) from which parameters can yield values in such way that it reduces the worst-case deviation from the best solution for processing time uncertainty represented by intervals (Rajba, 2021).

1.5 Algorithms for Solving Scheduling Problems

Problems with machine scheduling can be resolved using several solution algorithms. Three main algorithms are exact algorithms, heuristics, and metaheuristics, which are discussed in this section.

The exact algorithm is the only approach that guarantees an optimal result. However, with the dimension of the problems expanding, the time needed to resolve them grows substantially, and these exact algorithms may be insufficient to address massive situations. Among the many exact solution algorithms are Mathematical Programming (MP) and

Branch and Bound (B&B). The MP model is among the most significant algorithms (Najafi et al., 2016). The building of a basic mathematical model that will optimize specified criteria is represented by the scheduling model and customized for other types of scheduling environments. This includes job shops, parallel machines, flow shops, and single-machine scheduling. Notably, the use of MP in scheduling problems is common either alone or as part of an algorithmic framework. Scheduling problems with MP formulations are stepping stones to computational advances for some complex problems (Tom, 2019). An example of linear programming application in scheduling problems, such as those conducted by Hasan and Arefin (2017), presented four different real-life oriented scheduling models. This includes the scheduling of police, restaurant staff, bus drivers, and nurses.

The B&B algorithm depends on the suggestion that the total number of possible solutions can be broken up into smaller groups. Consequently, these smaller subsets can be assessed one by one until the best solution is determined, and further splitting gives lower and upper limits. Notably, the B&B algorithm can be used to obtain the best solution for individual problems (Pei et al., 2020). In addition, the B&B algorithm can be combined with other algorithms to form a hybrid algorithm for addressing the Flow Shop Scheduling Problem (FSSP) (Kumar et al., 2018).

Because of the complexity of many scheduling problems, heuristic algorithms are preferred over exact algorithms, which may not provide a solution in a reasonable period or could be costly to the decision-maker. Thus far, heuristics are problem-dependent algorithms that guarantee the best solution. This is compared to the optimal one since they are based on information discovery and learning that are frequently overly greedy and

become stuck in a local optimum (Tein, 2015). Such algorithms are categorized as heuristic and metaheuristic and are utilized to solve the UPMSP. Moreover, extensive research indicates that these algorithms are employed in combination to enhance one another (Đurasević & Jakobović, 2021). Dispatching Rules (DRs), a specific type of heuristic, incrementally build the schedule by allocating jobs to available machines according to their priority functions. Accordingly, the DR selects the subsequent job to be scheduled based on their rank. i.e., which job must be operated on which machine. Although they have limited knowledge of the problem, which impacts their performance (Ulaga et al., 2022), they can generate the schedule very quickly, making them suitable for problems with dynamic scheduling. Various DRs have been defined over the years for various scheduling criteria (Đurasević & Jakobović, 2018). Such heuristics have the advantage of being fast and straightforward, making them applicable to significant problems. At the same time, the neighborhood search algorithm is a widely recognized heuristic algorithm that offers near-optimal solutions. It is increasingly being used to tackle complex MSPs due to its efficient solution generation and ease of implementation. Hence, by applying it as a single or hybrid algorithm, it is feasible to ascertain near-optimal or optimal solutions. This approach improves the performance of other algorithms by intensifying the exploitation of promising regions in the search space and improving the quality of the results using neighborhood operations (Durasevi and Jakobovi, 2021). Additionally, several heuristic algorithms for UPMSP have been suggested over time, and a comprehensive literature review of UPMSPs can be referred to in Durasevi and Jakobovi (2021). In particular, this review focuses on solving machine scheduling using heuristic and metaheuristic algorithms.

Metaheuristics are a type of heuristic that guides and searches for reasonable solutions by combining fundamental heuristics with higher-level procedures. It is common practice to exchange efficiency to guarantee optimality when computing optimal solutions in real-world situations. In other words, by employing metaheuristic algorithms, the promise of discovering optimal answers is traded in favor of obtaining extremely decent results quickly. Many problems can be addressed effectively and professionally using metaheuristic algorithms created to solve NP-hard optimization problems with satisfactory results in an acceptable time (Dokeroglu et al., 2019).

Metaheuristic algorithms like Simulated Annealing (SA), Tabu Search (TS), and Genetic Algorithm (GA) are usually applied to solve MSPs. Observations of nature were the basis for most metaheuristic algorithms. Although a heuristic algorithms can yield good results in a challenging optimization problem, there is no assurance that the optimum results will be identified (Yang et al., 2016). Swarm Intelligence Algorithms (SIAs), Evolutionary Algorithms (EAs), and single solution algorithms are all types of metaheuristic algorithms that is suitable for a variety of combinatorial problems (Arık et al., 2022). Artificial Fish Swarm Algorithm (AFSA), for instance, classified under SIAs known as the biologically inspired metaheuristic, has proven effective in solving single-objective and MOO problems (Almshhadany & Ibrahim, 2018). The advantages of AFSA enable its efficient application in a wide range of optimization problems (Tan & Mohamad-Saleh, 2020). Therefore, the investigation of an AFSA in UPMSP can be conducted due to the limited applications of AFSA in uncertainty envioument. It was only used in the FSSP when Tirkolaee et al. (2020) suggested a hybrid AFSA with multi-objective fuzzy MP. Even though several studies attempted to solve UPMSPs using metaheuristics, no study

attempted to utilize the AFSA both in crisp and fuzzy UPMSPs to minimize makespan and tardiness.

Regardless of the type of solutions, none of the existing AFSA algorithms claim to have excellently solved all relevant optimization problems, and it continues to be an intriguing subject for many researchers.

1.6 Problem Statement

The uncertain nature of the MSP causes the associated data, including processing times and due dates, to become uncertain or not precisely quantifiable, potentially leading to misleading solutions. Hence, many studies incorporate uncertainty handling approaches, including stochastic, fuzzy, and robust scheduling, to address the unpredictable nature of scheduling parameters in MSP. However, when there is limited understanding of how these parameters vary and when historical data are unavailable, fuzzy representation or a range of values is considered more appropriate (Behnamian, 2016). Therefore, fuzzy set theory is employed to model uncertain parameters by incorporating triangular fuzzy numbers (TFNs), as they represent one of the simplest ways to express fuzzy numbers through membership functions. TFNs are used to represent both processing times and due dates to revise the UPMSP model proposed by Kongsri and Buddhakulsomsiri (2020), which minimizes makespan and total tardiness. In addition, the proposed model is simplified by removing the setup time constraint from the original model, making it more flexible and expandable.

The expandable model developed for large-scale UPMSP becomes computationally expensive and memory intensive due to its NP-hard nature. Meanwhile, metaheuristic

algorithms are promising for exploring new solutions and improving performance, owing to their flexibility and ability to search a wider solution space. Among these algorithms, the AFSA is a powerful continuous algorithm designed for solving NP-hard problems, featuring a multi-search strategy, strong global search capability, and applicability across various fields (Wang et al., 2016). However, the standard AFSA faces limitations when applied to discrete UPMSP. Therefore, instead of discarding this powerful algorithm, the continuous AFSA can be modified to solve the proposed discrete fuzzy multi-objective UPMSP model.

Although the effectiveness of the proposed modified AFSA algorithm was insufficient in some test instances, further improvements were required. The proposed modified algorithm performed less effectively when solving small-sized problems in instances four to ten, making it difficult to identify the best solutions and indicating that improvements to the algorithm remained challenging during the search process. Therefore, a hybrid AFSA algorithm was developed to enhance the performance of the modified AFSA by combining it with a neighborhood search algorithm, with a focus on balancing exploration and exploitation. Since the neighborhood search algorithm was designed for exploitation, it started from a single solution and iteratively explored its immediate neighbors to find improved solutions. This approach allowed frequent exploitation of the search space to achieve more accurate results while leveraging the strengths of the neighborhood search algorithm through neighborhood operations.

1.7 Research Questions

The following research questions are highlighted, considering the problem identification and discussion presented above.

1- What is the appropriate model to solve discrete UPMSP problems due to uncertain processing time and due date?

2- How can the continuous AFSA be modified to solve discrete UPMSP model in a fuzzy enviornment efficiently?

3- Is there any way to improve the performance of the modified AFSA algorithm?

4- How to measure the performance of the modified and hybrid algorithms?

1.8 Research Objectives

The main purpose of this study is to develop an effective modified and hybrid algorithm of AFSA, which is capable of managing multi-objective UPMSP with fuzzy set theory. This study also aims to improve the optimization procedures of fuzzy scheduling problems by providing a comparative analysis with optimization algorithms. In accomplishing this main objective, four specific objectives must be fulfilled, which are:

1- To revise the linear multi-objective model of Kongsri and Buddhakulsomsiri (2020) for UPMSP by embedding fuzzy parameters.

2- To modify the standard continuous AFSA to deal with the discrete UPMSP model in solving the proposed fuzzy UPMSP.

3- To construct a hybrid algorithm of the modified AFSA with the neighborhood search algorithm to improve the solution of the proposed fuzzy discrete UPMSP.

4- To assess the performance of the proposed algorithms in terms of minimum solution values by carrying out computational tests and comparing their results using different metaheuristic algorithms.

1.9 Scope of the Study

This study focuses only on solving a fuzzy multi-objective UPMSP involving two primary and conflicting objectives: minimizing makespan and minimizing total tardiness. The research adopts and extends the model developed by Kongsri and Buddhakulsomsomsiri (2020). The extension includes removing the sequence-dependent setup time constraint and incorporating TFNs to represent uncertainty in key parameters, namely processing times and due dates.

The methodological scope is restricted to the development of a modified AFSA specifically adapted for the discrete solution space of the UPMSP. The modification introduces new behavioral strategies, improved parameter mechanisms, and a transformation method designed to convert AFSA's continuous movements into valid discrete scheduling actions. In addition, this study focuses on developing a hybrid algorithm by integrating the modified AFSA with three neighborhood search operators, i.e swap, insertion, and reversion to better balance global exploration and local exploitation. The study does not include other swarm intelligence variants or hybridization frameworks beyond those specified.

The evaluation of the proposed algorithms is limited to simulated test instances of varying sizes executed within a MATLAB environment. Comparisons are conducted only against selected modified AFSA variants and established metaheuristic algorithms. The

performance evaluation used in this study are restricted to minimum, maximum, mean, standard deviation, and the Wilcoxon signed-rank test to determine statistical significance. Importantly, this research focuses exclusively on solution quality and algorithmic development. Although computational time is recognized as an essential performance evaluation in optimization studies, it is explicitly excluded from comparative evaluation due to its variability across different hardware specifications, programming environments, and implementation structures.

1.10 Significance of the Study

This research contributes to the field of scheduling optimization by addressing the uncertainty and complexity commonly encountered in real-world environments. In practical settings, scheduling decisions are often influenced by unanticipated events such as inaccuracies in parameter estimation, fluctuating weather conditions, workforce health, machine breakdowns, and various human factors. By incorporating fuzzy multi-objective considerations, this study provides a more realistic and adaptable framework for handling such uncertainties.

The proposed methodology enhances scheduling efficiency by producing acceptable and reliable solutions within time, cost, and quality constraints. Focusing on the UPMSP, this research emphasizes the importance of UPMSP both as a general theoretical model and as a practical tool applicable across diverse industries and operational environments.

A key significance of this study lies in the development of new modified and hybrid algorithms inspired by the AFSA to effectively solve the fuzzy multi-objective UPMSP. The hybrid integration of global exploration (AFSA) with strong local exploitation

(neighbourhood search) provides a more powerful solution framework for complex scheduling problems.

Furthermore, the methodology and algorithmic strategies presented in this study are not limited to UPMSP alone; they can be extended to other multi-objective optimization and machine-environment problems. Consequently, this research broadens the understanding of scheduling under fuzzy uncertainty and serves as a valuable reference for future studies involving parallel machine problems and multi-objective optimization in uncertain environments.

1.11 Structure of the Thesis

This thesis consists of six chapters. The following are brief reviews of these chapters:

Chapter One describe the context and background of the study of MSP. In addition, this chapter presents the problem statement, research questions, and objectives, and summarizes the methods for solving scheduling problems. This chapter also outlines the scope and significance of the study.

Chapter Two reviews the literature on different MSPs in a fuzzy environment and provides an overview of MOO algorithms categorized as exact, heuristic, and metaheuristic. This is conducted by summarizing previous research and analyzing currently available literature in the relevant area. The hybridization algorithms for solving MSPs are followed by a description of the five most relevant previous studies as a similar study. Following that, the procedures of standard AFSA and a modified AFSA for solving UPMSP are presented and reviewe and research gaps are explained.

Chapter Three started by describing five phases that can be used to provide an overview and explanation of the current study's methodology and the research process. This is followed by a description of the problem, identification of the parameters, and formulation of the mathematical model. Subsequently, this chapter identifies and describes the fuzzification process, the fuzzy set theory definitions and defuzzification method of the proposed UPMSP problem.

Chapter Four elaborates on the proposed solution algorithms, which consist of three phases: the details of the development of the proposed modified AFSA and the hybrid AFSA methodology.

Chapter Five covers the computational, experimental results, and analysis of the proposed modified and hybrid AFSA in UPMSP in a fuzzy environment to minimize multi-objective makespan and total tardiness. The performance of the proposed modified AFSA algorithm is compared to other existing modified AFSA algorithms, and the hybrid AFSA is also compared to well-known metaheuristic algorithms.

Chapter Six discusses the research's contribution and limitations and outlines the directions for future research.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction to Machine Scheduling Problem

This chapter offers a detailed review of the literature on this research, beginning with the background of the Machine Scheduling Problems (MSPs) in a fuzzy environment. This remains through many related works of literature that utilize several algorithms to solve MSPs, as well as fuzzy machine problem formulation, to provide motivation for this study. It continues with determistic environment to study the methods to solve MSPs, such as optimization algorithms, heuristics, metaheuristics, Swarm Intelligence Algorithms (SIAs), and hybridization algorithms. Subsequently, the standard artificial fish swarm algorithm (AFSA) and a modified AFSA for solving unrelated parallel machine scheduling problems (UPMSP) are presented and reviewed. Finally, similar study and chapter summary are revisited and discussed.

2.2 Machine Scheduling Problems in a Fuzzy Environment

Since the early 1950s, there has been a substantial quantity of literature on scheduling in the discipline of operations research. MSP is a significant discrete optimization problem with the main objective of efficiently allocating shared resources to activities over time, where resources are machines and jobs are activities (Gomes, 2017).

As a result of the immense effect of MSPs on different aspects of people's lives in many areas of real life, they significantly boost manufacturing system productivity, whether through their use in the transportation industry, the production industry, or workforce distribution. The majority of MSPs research in recent years has been under the assumption

of a well-defined and precise set of parameters for the problem at hand (deterministic models). Unfortunately, various levels of uncertainty are often involved in manufacturing processes, which directly affect how smoothly the process runs.

The variation among the amounts of knowledge required to accomplish a task and the amounts of available knowledge is known as uncertainty (Xiaoyue, 2020). Uncertainty can be modeled in three ways; stochastic scheduling, fuzzy scheduling, and robust scheduling according to the nature of the problems (Rajba, 2021). A fuzzy representation or a range of values can be used when there is a poor understanding of how the parameters vary and no historical data exists (Behnamian, 2016). In this context, it is essential to include the model of uncertainty in the scheduling problems. In this situation, when there are uncertainties in the problem's parameters, the objective functions are also uncertain. In 1965, Zadeh pioneered mathematically representing fuzzy set theory of imprecision or ambiguity inherent in actual scheduling problems in everyday life.

Researchers have long been interested in scheduling problems involving fuzziness, as evidenced by numerous studies on fuzzy machine scheduling, such as those proven in a survey by Behnamian (2016) and Abdullah and Abdolrazzagh (2014). The mathematical branch known as Multi-Objective Optimization (MOO) tackles the multi-objective decision-making of challenging optimization problems (Lin & Gen, 2018). Multiple functions need to be enhanced simultaneously in optimization problems with multiple targets (Ojstersek et al., 2020). Furthermore, uncertainty and MOO must frequently be considered in tandem with real-world manufacturing applications. Previous researchers have employed various methodologies and algorithms to solve this uncertainty and multi-objective scheduling problem. This ranges from exact techniques to heuristic algorithms in single and multiple-machine scenarios. Therefore, in order to cover as many recent

studies as possible that have dealt with machine scheduling in a fuzzy environment for different objective functions, this section review the literatures on machine scheduling for single, parallel, job shop, and flow shop problems.

2.2.1 Fuzzy Multi-Objective Scheduling on a Single Machine

The single MSP is a significant decision problem with one or more objectives that must be considered at once, where there is just one machine available ($M = 1$), and there is only one operation per task ($O_j = 1, j \in J$). A single MSP is provided by Jayanthi and Karthigeyan (2017), optimizing the sum of weighted earliness and tardiness costs by utilizing trapezoidal and TFNs for processing times and due dates, respectively. Meanwhile, Khalifa (2020) examined a single-machine problem that utilized interval-valued fuzzy numbers as due dates to reduce the cost of an earliness and tardiness penalty function in a fuzzy environment. In the meantime, Durmaz et al. (2021) minimize maximum tardiness and earliness as well as total weighted tardiness, making the fuzzy measure an effective way of integrating multiple objectives in a single MSP. Meanwhile, the fuzzy programming model developed by Jadhava et al. (2024) for a multi-objective single machine scheduling problem with minimizes the total weighted tardiness and the total weighted completion time simultaneously, where job processing time is considered as an independent normal random variable. The computational findings have been stated to demonstrate the effectiveness of the proposed approach.

Although single-machine scheduling has received the most attention in research since it is easier to solve, it is not practical in manufacturing environments. This includes

mechanical, electronic, and other industries, where the complexity usually grows with the number of machines.

2.2.2 Fuzzy Multi-Objective Scheduling Problems on Parallel Machines

One of the fundamental and notable concerns that management faces is the proper scheduling and sequencing of work on machines. Many problems that are relatively easy to solve in a single-machine environment become Nondeterministic Polynomial-Time hard (NP-hard) when extended to parallel machine settings (Pavlov et al., 2020). It means that, there are several machines ($M > 1$) and just one operation per job ($O_j = 1, j \in J$), indicating that each machine performs the same function. Parallel Machine Scheduling Problems (PMSPs) arise in many sectors, ranging from manufacturing systems to information technology environments such as parallel processing and distributed computing. As a result, PMSPs have become one of the most widely studied topics in the scheduling literature (Rostami et al., 2015). In addition to the different PMSP settings, various optimization objectives are frequently considered in machine scheduling. These objectives are typically associated with the decision-making processes of manufacturers or service providers. Common objectives include minimizing the makespan, which reflects production efficiency, and minimizing total tardiness, which relates to meeting customer due dates and improving service quality (Rostami et al., 2015).

In essence, concurrently considering total tardiness and makespan minimization could lead to increased effectiveness. Although this multi-objective is commonly used for scheduling machines (Guevara-Guevara et al., 2022), the information needed to specify the time constraints and conditions maybe vague or not precisely measurable. To improve the quality of the schedule and make it more reflective of real scheduling issues, fuzzy set

theory can be used to evaluate UPMSPs. Nondeterministic parameters, like processing times and due dates, can be expressed with fuzzy numbers to denote uncertainty and ambiguity. However, other variables also affect a problem's parameters, such as changing the order's due date, the weather, traffic congestion, the condition of the driver, and numerous others. Every UPMSP has various types of machines, proving that not all are similar. These changes will result in varying machine processing speeds when performing tasks (Rostami et al., 2015). Additionally, the objectives, constraints, preferences, and client expectations that are associated with humanism are frequently vague and are subject to change as time passes. Hence, when the fuzzy set theory is applied to the machine setting, the scheduling problems become more complicated.

Fuzzy machine scheduling has received a lot of attention from industry and academia, and numerous objectives have been thoroughly studied. For example, by employing TFNs for parameters associated with due dates and processing times, Torabi et al. (2013) developed a fuzzy programming UPMSP model designed to minimize the total weighted lateness and flow times, along with the variation in machine load. However, Naderi-Beni et al. (2014) used Trapezoidal Fuzzy Numbers (TrFNs) to present processing times, release dates, setup times, and due dates to create a fuzzy multi-objective Mixed Integer Linear Programming (MILP) model for UPMSPs that aimed to reduce simultaneously workload imbalance and total tardiness. In contrast, Behnamian (2014) employed a centroid defuzzification function and bell-shaped fuzzy processing times to rank fuzzy numbers and compute completion times. With the processing time signified as TrFNs, Yeh et al. (2014) presented a uniform PMSP that minimizes makespan by considering learning effects and processing time uncertainty. A multi-objective mixed integer nonlinear Mathematical Programming (MP) model was introduced by Rostami et al. (2015) to

identify a Pareto optimal front of makespan and total earliness/tardiness when processing times and due dates are provided as TFNs in uniform PMSP. Furthermore, Nailwal et al. (2015) managed scheduling on fuzzy parallel machines to reduce maximum tardiness and weighted flow time. Meanwhile, Asghari and Nezhadali (2016) presented a fuzzy multi-objective mathematical approach that simultaneously minimizes the totals of weighted tardiness, earliness, flow time, and the cost of machine deterioration with fuzzy parameters to solve the proposed model.

Using TrFNs to indicate processing, release, and setup times, Liao and Su (2017) developed the fuzzy UPMSP. They used a fuzzy ranking method to rank fuzzy makespan and other fuzzy numbers. At the same time, TFNs were utilized by Manupati et al. (2017) to represent processing times and deadlines for fuzzy multi-objectives to minimize machine-load variation, tardiness, makespan, and flow time. Moreover, the fuzzy linguistic method is proposed by Geyik and Elibal (2017) for the non-identical PMSP using fuzzy processing times as TFNs to measure the average makespan. To minimize the cost of setting due dates and penalties for tardiness and earliness, Arık and Toksarı (2018) presented a fuzzy MP model which employs TFNs to signify job times, learning effects, and deterioration coefficients. To describe the multi-objective parallel machines problem, Salimifard et al. (2019) examined TFNs and presented a fuzzy mixed-integer goal programming model with the objective of minimizing hazardous waste and total tardiness. For unrelated parallel batch scheduling, where each job is determined by TrFNs of processing time-dependent machines, ready time, and due date, Sadati et al. (2019) suggested a fuzzy multi-objective model to minimize makespan and maximum tardiness. Notably, makespan has been reduced to uniform PMSP by Li et al. (2019) when processing times and resource consumption are both TFNs. In addition, Jia et al. (2019)

examined schedulers on parallel batch processing machines of various capacities and fuzzy processing times indicated as TFNs to minimize the makespan. Using TFNs' setup, processing times, and trapezoidal due dates, Rezaeian et al. (2020) created a fuzzy MP model to reduce overall fuzzy weighted earliness and tardiness penalties in UPMSPs. Similarly, fuzzy makespan minimization is addressed by Li et al. (2020) employing a uniform PMSP that incorporates deterioration and learning effects, as well as job processing times represented as TFNs. For non-identical PMSP, Wang et al. (2021) suggested using TFNs to exhibit processing times and minimize makespan. Using a fuzzy trapezoidal due date, Xu et al. (2021) created a multi-objective, non-identical parallel model that aims to minimize makespan and costs related to tardiness and earliness. In the field of textile manufacturing, Çini et al. (2022) evaluated how to minimize the makespan using TFNs to represent the processing and setup times on the same PMSP.

To address UPMSPs, Yaghtin and Javid (2023) created a fuzzy-based genetic algorithm that minimizes total tardiness with periodic maintenance and process constraints like setup time, release date, and processing times and due dates are TFNs. The algorithm show good performance for large instances based on quality and run time factors. Meanwhile, Srivastava and Singh (2023) presented an evolutionary strategy that employs a combination of greedy and random heuristics to reduce total weighted completion time in UPMSP with sequence and machine-dependent setup times, where processing times are represented as TFNs. When compared to the state-of-the-art approach, the proposed approach produces better results in significantly less time.

Among the PMSPs reviewed, UPMSPs have special significance due to their widespread use in many industrial applications. As a result, researchers and decision-makers are still

working on identifying an optimal schedule. Hence, continuous improvement is required to improve solution approaches and meet the evolving requirements of UPMSPs.

2.2.3 Fuzzy Multi-Objective Scheduling Problems on Job Shop

In a Job Shop Scheduling Problem (JSSP), the number of operations per job might vary, as well as the type of jobs and their sequences. A machine might perform a job more than once or not be used at all. Additionally, every operation can only be performed on one machine.

In solving fuzzy JSSP, Xu et al. (2018) used TFNs to indicate processing time to reduce fuzzy makespan and thus partially solve fuzzy flexible JSSP. Abdolrazzagh and Sarbishegi (2019) introduced a multi-objective fuzzy JSSP with a fuzzy due date, minimizing the makespan's maximum and maximizing the satisfaction degree's minimum. In addition, a fuzzy multi-objective based on completion time and the anticipated due date is introduced by Gao et al. (2020) in a fuzzy JSSP where a job's start time, processing time, and completion time are all TFNs. Li et al. (2022) presented the MILP model for a flexible JSSP with processing times as TFNs to minimize the makespan and the total workload simultaneously. Meanwhile, Afsar et al. (2022) considered multi-objective fuzzy JSSP with processing time and starting time of jobs as TFNs to minimize makespan and non-processing energy. A fuzzy JSSP model that minimizes makespan is also presented by Wu et al. (2025), TFNs are used to represent the uncertainty related to processing and transportation time in this model, which is solved by an improved discrete particle swarm optimization (PSO) algorithm. The results of the comparison experiment indicate that the suggested algorithm performs competitively when compared to other selected efficient algorithms. Since JSSP includes running a set of N jobs on a set of M machines, any job consists of a set of operations, and any operation

is processed by a machine. Hence, it is considered one of the most complex problems, both theoretically and practically, in the field of scheduling (Peng et al., 2015).

2.2.4 Fuzzy Multi-Objective Scheduling Problems in a Flow Shop

Flow Shop Scheduling Problems (FSSPs) are widely researched. The number of operations in each job is the same ($O_J = C, j \in J$), and each job uses identical orders for the machines. In addition, each job can have the first machine perform the initial operation, the second machine perform the second operation, and so forth. These processes are also known as stages. In many cases, fuzzy set theory was successfully implemented to solve FSSPs, which involve scheduling jobs on machines with fixed processing times to achieve a preset-specific objective. In a fuzzy environment, Najari et al. (2018) considered multi-objective FSSPs to minimize the makespan, job tardiness, and tardiness cost and maximize net present value. Meanwhile, Golneshini and Fazlollahtabar (2019) solved the MIP models with hybrid FSSPs with unrelated machines and fuzzy numbers for processing times and due dates to reduce overall completion time and maximum job lateness. Furthermore, TFNs indicate fuzzy processing time, as Cai and Lei (2021) presented for distributed hybrid FSSP, to simultaneously optimize fuzzy multi-objectives for makespan, total agreement index, and total energy consumption. On the same note, Engin and Yılmaz (2022) formulated a fuzzy model for processing time and the due date for FSSP with multi-objectives. In order to optimize the makespan, total machine load, and maximum machine load for the fuzzy job shop problem, Liu et al. (2023) developed a scheduling model which employs TFNs for processing times. The improved multi-objective evolutionary algorithm was proposed to be based on multi-strategy dynamic evolution. Next, a variable

neighborhood search technique is added to enhance the algorithm's search capabilities. The proposed algorithm's effectiveness was proven through experimental validation.

In the meantime, Ranjith and Karthikeyan (2024) introduced a fuzzy FSSP with job processing times represented by TrFNs to reduce total job waiting time. A novel method is developed to solving the scheduling problem under investigation, and the obtained results highlight the effectiveness of the proposed method when compared to existing algorithms.

Following the review of the four primary MSPs in fuzzy environments, which covered numerous recent studies, it is crucial to study the techniques used to solve these problems using different algorithms.

2.2.5 Solution Algorithms for Solving Machine Scheduling Problems in a Fuzzy Environment

MSPs can be solved using various techniques for dealing with fuzziness and uncertainty. Most problems with scheduling naturally take the form of multi-objective problems. Thus, various algorithms are utilized to optimize multi-objective problems. For instance, in the context of the single MSP, Jayanthi and Karthigeyan (2017) employed a Firefly Algorithm (FA) and conducted the comparison with PSO algorithm. Khalifa (2020) employed a particular technique that helps identify the optimal solution for small systems. Meanwhile, Durmaz et al. (2021) presented a Genetic Algorithm (GA) aimed at achieving near-optimal solutions to the issue.

For identical PMSPs, Behnamian (2014) suggested a Discrete Particle Swarm Optimization (DPSO) to minimize the fuzzy makespan when addressing PMSP. The findings indicate that the suggested algorithm is highly effective for various structural instances compared to the lower bound. Nailwal et al. (2015) suggested a heuristic

algorithm for determining the best possible order, and the computations indicate the efficiency of the suggested algorithm. Moreover, Arık and Toksarı (2018) presented a local search algorithm to produce efficient solutions for large-scale problems within a reasonable time frame. A GA-based solution approach was presented by Salimifard et al. (2019), and the numerical results reveal how successfully the solution algorithm identifies the most preferred solution. Meanwhile, Jia et al. (2019) proposed the Fuzzy Ant Colony Optimization (FACO) algorithm, which proves outstanding outcomes when compared to other algorithms in a reasonable time. Xu et al. (2021) developed a hybrid approach that combines a Grey Wolf Optimizer (GWO) and an Evolutionary Algorithm (EA). The experimental findings suggest that the suggested algorithm outperforms three existing MOO algorithms. Furthermore, Çini et al. (2022) created a randomized search algorithm and fuzzy MILP formulation to overcome the problem they proposed.

For uniform PMSPs, Yeh et al. (2014) proposed two approaches, the Simulated Annealing (SA) algorithm and GA, to solve problems that involve computational studies, which were conducted to evaluate the performance of heuristics in a variety of settings. The study reported that the SA outperformed the GA. Rostami et al. (2015) presented an MP model and a Branch and Bound (B&B) algorithm. Though, the computational results indicate that it struggles to solve large problems in a logical running time. Li et al. (2019) demonstrated that Simplified Swarm Optimization (SSO) outperforms GA and PSO based on research findings. A Simplified Swarm Optimization Local Search (SSOLS) was presented by Li et al. (2020). The results revealed that the proposed algorithm offers acceptable solutions in a reasonable time and outperforms the compared algorithms. In addition, the Improved Fruit Fly Optimization Algorithm (IFOA) served as the foundation

for Wang et al.'s (2021) improved intelligent heuristic, which consistently outperforms and yields better results.

For UPMSPs, Torabi et al. (2013) developed an efficient multi-objective particle swarm optimization (MOPSO) algorithm for UPMSP. Based on tests, the proposed algorithm outperforms the conventional multi-objective particle swarm (CMOPSO) in all three performance metrics. In contrast, Naderi-Beni et al. (2014) employed two metaheuristic techniques, Fuzzy Non-Dominated Sorting GA (FNSGA-II) and fuzzy multi-objective particle swarm optimization (FMOPSO), to address the proposed model. FNSGA-II outperforms FMOPSO in terms of efficacy, particularly for large-scale problems. At the same time, Asghari and Nezhadali (2016) formulated a heuristic approach depending on GA, and the computational results validated the effectiveness and efficiency of the proposed heuristic in addressing time complexity. In addition, Geyik and Elibal (2017) proposed a heuristic method that takes into consideration the longest processing time for solving a specific problem and assesses the average makespan. Liao and Su (2017) developed a hybrid Ant Colony Optimization (ACO), demonstrating superior performance compared to a hybrid PSO and two variants of SA algorithms. Manupati et al. (2017) employed two metaheuristic techniques to create effective solutions for the UPMSP model: Artificial Immune Systems (AISs) and conventional Non-Dominated Sorting GA II (NSGA-II). The performance of multi-objective PSO in UPMSP was compared, and excellent results were obtained. Consequently, Sadati et al. (2019) proposed two exact methods (two-phase fuzzy and ε-constraint), along with two metaheuristic Fuzzy Multi-Objective Discrete Teaching-Learning-Based Optimization (FMODTLBO) and FNSGA-II. The results revealed that the FNSGA-II performed slightly better than the FMODTLBO. Finally, two metaheuristic algorithms, modified SA

and GA, were created by Rezaeian et al. (2020). A comparison revealed that modified SA performed better than GA. The significance of the studies presented in such investigations can be focused on PMSPs. As a result, this section focuses on the methods used to solve PMSP problems, which are the subject of this study. Table 2.1 summarizes research on fuzzy set theory for scheduling parallel machines, covering fuzzy parameters, objectives, parallel machines, and algorithms.

Table 2.1

Review of research on fuzzy set theory for scheduling parallel machines

Authors	**Fuzzy parameters**	**Objective**	**Parallel machine**			**Algorithms**		
			Identical	Uniform	Unrelated	Exacts	Heuristic	Metaheuristic
Çini et al. (2022)	P_j and *sijk* as a TFNs	Fuzzy makespan	√				Randomized search algorithm	
Xu et al. (2021)	d_j given as TrFN	Minimize makespan and earliness/tardiness penalty cost	√					GWO
Wang et al. (2021)	P_j is TFNs	Minimize makespan		√			IFOA	
Li et al. (2020)	P_j is TFNs	Fuzzy makespan		√				SSOLS
Rezaeian et al. (2020)	P_j and *sijk* as a TFN, d_j given as TrFN	Minimize total weighted fuzzy earliness and tardiness			√		GA & SA	
Li et al. (2019)	P_j and resource constraints are TFNs	Fuzzy makespan		√				SSO algorithm
Jia et al. (2019)	P_j is TFNs	Minimize makespan	√				FACO	

Salimifard et al. (2019)	P_j and *sijk* as a TFNs	Minimize the total tardiness and the total hazardous waste	√				GA	
Sadati (2019)	P_j , d_j, and r_j are given as TrFN	Makespan and maximum tardiness			√	two-phase fuzzy & ε-constraint		FNSGA-II & FMODTLBO
Arık & Toksarı (2018)	P_j, LE, and DE are TFNs	Tardiness, earliness, and cost setting	√				local search algorithm	
Manupati et al. (2017)	P_j & d_j are given as TFNs	Minimize makespan tardiness, flow time, and machine-load variation			√			CNSGA-II & AI-NSGA-II
Geyik & Elibal (2017)	P_j is TFNs	Minimize makespan			√		a heuristic algorithm	
Liao & Su (2017)	P_j, r_j, and *sijk* as (TrFN)	Minimize makespan			√			FHACO
Asghari & Nezhadali (2016)	P_j & d_j are given as TFNs	Minimize Four objectives			√		GA	
Rostami et al. (2015)	P_j & d_j are given as TFNs	Total earliness /tardiness (ET) and makespan		√		MINLP & (B&B) algorithm		

Nailwal et al. (2015)	P_j is TFNs	The weighted flow time and maximum tardiness	√				a heuristic algorithm	
Behnamian (2014)	bell-shaped fuzzy processing times	Fuzzy makespan	√					DPSO
Naderi-Beni et al. (2014)	P_j , r_j, d_j, and *sijk* are given as TrFN	Minimize workload imbalance and total tardiness			√			FMOPSO & FNSGA-II
Yeh et al. (2014)	P_j *is* given as TrFN	Fuzzy makespan		√			GA & SA	
Torabi et al. (2013)	P_j & d_j are given as TFNs	Minimize five objectives			√			MOPSO

According to the previous literature review discussed in Table 2.1, it can be observed that the utilization of exact methods is comparatively limited when compared to other solution methods. TFNs are the most commonly used fuzzy parameters, whereas unrelated parallel problems are the most frequently studied. However, makespan and tardiness are frequently addressed as a single or multi-objective problem. Since UPMSP is NP-hard, it adds complexity to determining a best solution in terms of minimum MOO problems in a fuzzy environment. Consequently, numerous new modified or hybrid metaheuristics have been introduced in recent years to deal with this problem.

In the context of JSSP, several metaheuristics have been developed. For instance, Li et al. (2022) presented a hybrid EA to address fuzzy multi-objective difficulties. Afterward, Afsar et al. (2022) applied a new enhanced memetic algorithm to solve their proposed problem. Gao et al. (2020) presented an innovative selection mechanism designed to improve the generic differential evolution algorithm, aiming for superior optimization results. Moreover, Abdolrazzagh and Sarbishegi (2019) developed a modified PSO to achieve the optimal pareto solutions. In addition, Xu et al. (2018) presented a metaheuristic optimization technique known as the Flower Pollination Algorithm (FPA). It is inspired by the natural pollination process among flowering plants to solve their proposed problem.

Finally, in the FSSPs, Engin and Yılmaz (2022) presented an effective GA intended to solve the fuzzy problems that were formulated. Cai and Lei (2021) utilized the shuffled frog-leaping algorithm to enhance search efficiency. Golneshini and Fazlollahtabar (2019) also introduced PSO and GA. In addition, Najari et al. (2018) introduced two EAs, namely the NSGA II and the non-dominated ranked GA, to identify Pareto solutions.

To sum up, the preceding review indicates that TFNs are the most common fuzzy parameters used and that the primary objectives examined, whether single or multi-objective, are tardiness and makespan. After a review of some research on fuzzy set theory for MSPs, this research proceeds to discuss the solution algorithms.

2.3 Existing Algorithms for Solving Machine Scheduling Problems in Deterministic Environment

In deterministic MSPs, all parameters and constraints are fully defined and known in advance. Optimization algorithms therefore play a crucial role in addressing these scheduling challenges. This section introduces and reviews the primary solution approaches applied in deterministic scheduling, focusing on three major categories of algorithms: exact, heuristic, and metaheuristic algorithms.

2.3.1 Exact Algorithms

Exact algorithms deliver the optimum solution to the problem. The use of exact optimization is restricted compared to other algorithms since gaining optimal results in a reasonable time utilizing exact algorithms is prohibitively expensive. Furthermore, the challenge with using this class of algorithms for more difficult optimization problems, i.e., NP-hard problems, is a significant disadvantage. Mathematical models offer a framework for tackling most schedu ling issues to yield optimum or nearly optimum results for different performance measures. Research related to the schedule has been published using a mathematical model to help management. Various methods, including integer, mixed integer, goal, and linear programming, have been developed and successfully adopted to maximize or minimize problems with single or multiple objectives. The lowest-

cost scheduling problem solutions are also possible using mathematical optimization methods with a set-covering formulation (Noor et al., 2022).

Kongsri and Buddhakulsomsiri (2020) proposed a Mixed Integer Linear Programming (MILP) model that minimizes multi-objective makespan and total tardiness in UPMSP with sequence-dependent setup time constraints. The results produced optimal solutions for small-sized problems involving only three machines and ten jobs. As a result, the exact algorithm lacks the flexibility to find the best solutions within a reasonable timeframe for large problems, particularly when time constraints are present.

Hence, the B&B algorithm is frequently applied to solving scheduling problems using the exact algorithms (Rifai et al., 2021). As a tree search, the B&B algorithm employs an implicit search space enumeration. The main concept of B&B is the division of a problem into easily solvable sub-problems (Soto et al., 2020). In addition, Malhotra et al. (2022) used the B&B algorithm for identical PMSP to simultaneously reduce the multi-objective total costs of tardiness and earliness and makespan with processing times expressed as TFNs. Varol and Subulan (2021) suggested an exact method to solve the flexible JSSP based on a fuzzy-stochastic MIP model of multi-objective total tardiness and makespan. At the same time, TFNs are used to represent fuzzy processing and setup times and stochastic due dates. Sadati et al. (2019) proposed two exact algorithms (two-phase fuzzy and ε constraint) that are employed to generate a set of Pareto solutions for the fuzzy multi-objective MILP model for a UPMSP to minimize makespan and maximum tardiness. In addition, TFNs and TrFNs are utilized to signify job processing times to minimize total elapsed time, as proposed by Kumar et al. (2018), who combined the B&B algorithm with GA to solve the fuzzy FSSP. Heydari and Aazami (2018) solved a multi-objective

makespan and maximum tardiness model simultaneously for the JSSP with the ε-constraint method. Consequently, a set of numerical data is created and examined to confirm the model's effectiveness and adaptability.

The benefit of these algorithms is that they assurance the best global solutions and can be used to solve problems that require high precision. However, it has slow convergence and can only solve minor problems. Thus, it cannot be used to solve complex problems (Xu et al., 2022). As a result, heuristic algorithms are frequently employed in MSPs to efficiently identify near-optimal solutions, particularly when exact algorithms become impractical due to the problem's difficulty.

2.3.2 Heuristic Algorithms

A fascinating instance of a problem involving combinatorial optimization is a scheduling problem, which is frequently easy to define yet hard to solve, as evidenced by the reality that many of them are NP-hard (Tahami & Fakhravar, 2022). Traditional optimization algorithms are ineffective for dealing with complicated, NP-hard problems with multiple conflicting objectives and high-dimensional, nonlinear, and hybrid problems in many real-world problems, such as scheduling (Rajabi Moshtaghi et al., 2021). Since these problems must be resolved, the only alternative is to accept local or sub-optimal solutions with an appropriate accuracy and optimization time. Consequently, heuristic algorithms are developed to quickly produce acceptable results based on some rules. Correspondingly, two constructive heuristics are presented by Shao et al. (2020) to reduce the fuzzy makespan when the processing time is signified as TFNs for the fuzzy FSSP. Based on knowledge of the specific problem, the first heuristic was proposed to produce the initial

job sequence using the spread value of fuzzy processing time, and the second reduces the total expected idle time and blocking time to assign partial jobs to factories. Furthermore, Ma et al. (2019) presented two heuristic algorithms for solving the uniform parallel machine with uncertainty and randomness to reduce the expected completion time after transforming the model into a crisp MP model using chance theory. Additionally, Kanakavalli et al. (2018) proposed a new fuzzy method for identifying solutions to the concurrent flexible manufacturing scheduling problem to reduce the makespan with the resources available. Among the most used heuristic algorithms with real-world applications is the neighborhood search algorithm, which is based on a simplex algorithm to identify nearly optimal solutions.

Neighborhood search algorithm is a general algorithm to developing heuristic algorithms for discrete optimization. In addition to being easy to understand and use, neighborhood search algorithm is unique since it is general. It also does not make any assumptions about the objective or constraints, while other heuristic algorithms depend on the specific problem at hand. Furthermore, the neighborhood search algorithm is designed for exploitation. It starts with a single solution and iteratively explores its immediate neighbors to find an improved one (Brandimarte, 1992). Note that the local improvement algorithm is fundamental to neighborhood search algorithms. The heuristic algorithms derived from neighborhood search remain popular due to their simplicity, efficiency, and ability to generate precise and customized options (Ning et al., 2015). The algorithm incorporates insert, swap, and sub-interchange operations to enhance the classical SA algorithm's capacity. It also offers better solutions in shorter runtimes to minimize tardiness and earliness objectives for flow shop problems (Karacan et al., 2023).

The neighborhood search algorithm employed by Liu et al. (2022) examined the iron-steel allocation problem in a steel company where time for initial treatment was ambiguous. Two operations, swap and insertion, are applied to minimize the weighted completion time while maintaining constraints with uncertain parameters. Conversely, a study by Mortada and Yusof (2021) mentioned that the use of three neighborhood search operations (two interchanges, cross-exchange, and 2-opt) for Artificial Bee Colonies (ABC) is effective for improving the local search operations of the swarm bees algorithm. This approves the optimization of the minimum total travel distance result and the number of vehicle routing problems with time windows. In essence, the algorithm yields the best outcomes compared to other optimization algorithms and can solve a wide range of problems.

Due to its good performance, instead of embedding the neighborhood search algorithm into a single algorithm, it is also incorporated into hybrid algorithms. For instance, Abdullah et al. (2021) proposed a hybrid meta-heuristic algorithm to solve the UPMSP to minimize the makespan using a set of neighborhood search operations (Shift, Two-Shift, Switch, Task Move, and Swap). These operations are designed to exploit the area around the solutions obtained by an evolutionary descent framework. Overall, the results of neighborhood experiments demonstrate that the algorithm outperforms the compared algorithms on several UPMSP instances.

Li and Yin (2013) developed a hybrid cuckoo search algorithm that integrates cuckoo search with several neighborhood operations as a local search strategy: inverse, insert, and swap. The aim was to enhance the population diversity and quality of solutions to solve

the flow shop problem and minimize makespan. Notably, the achieved experimental results prove the effectiveness and efficiency of the algorithm.

In heuristic algorithms, neighborhood operations are conducted to identify neighboring solutions. The best solution is possible when the defined neighborhood region covers the entire feasible region. Thus, choosing the suitable operations is critical to successfully exploring the search space (Karacan et al., 2023).

Neighborhood search algorithm is a wide-ranging algorithm for constructing heuristic algorithms for discrete optimization. They have been applied to a variety of optimization issues involving unrelated machines (Ulaga et al., 2022), Flexible Job Shop Scheduling Problems (FJSSP) (Serna et al., 2021), vehicle routing problems (Mortada & Yusof, 2021), and production scheduling problems (Lei & Guo, 2012). The definition of neighborhood search can take many forms. However, it is usually defined to ensure that every solution in the neighborhood of the current solution satisfies a predetermined set of requirements.

While heuristic algorithms search the solution space to find a good solution, metaheuristics is the concept of exploring the search space by implementing different strategies. These strategies are used to identify a high-quality solution in a short time within the regions in the search space. The following subsection explains the metaheuristic and the primary categories of evolutionary algorithms and swarm intelligence algorithms.

2.3.3 Metaheuristic Algorithms

Metaheuristics are algorithms that work independently of the problem and make it feasible to completely discover the solution space and, ideally, arrive at a superior solution. Start with complete solutions and iteratively enhance them by integrating modifications to search as far out of the potential solution space as possible. Since it attempt to enhance

the existing schedule rather than build it greedily, such strategies may produce very satisfying outcomes (Ulaga et al., 2022). Metaheuristic algorithms seek solutions that are as close as possible to the ideal. They usually repeat the process of generating or searching for an efficient solution (Ezugwu et al., 2021). Metaheuristic algorithms are classified in various ways based on the selected characteristics. Accordingly, metaheuristics are divided into the algorithms working on a single solution at any given time, a large class of improvement algorithms known as local search-based (neighborhood search) metaheuristic algorithms. In addition, they search the "neighborhood" of the existing solution to identify an improved solution at each iteration. Notably, they are highly successful and widely employed as heuristic algorithms in operations research (Ahuja et al., 2002). The main algorithms in this class are iterative local search, variable neighborhood search, greedy randomized adaptive search, scatter search, and Tabu Search (TS) (Afshari et al., 2019).

In contrast, population nature-inspired metaheuristic algorithms executed several operations iteratively and searched with multiple initial points to select the best candidate from a set of candidates (Ghaffar et al., 2022). This includes EAs and SIAs (Beheshti & Shamsuddin, 2013). Specifically, a equilibrium among exploration and exploitation abilities is a critical factor in population nature-inspired algorithms. Moreover, the success of EAs and SIAs arises from nature-inspired operations that may have a mechanism to adjust the two exploration and exploitation abilities (Nakane et al., 2020), which is discussed in the following subsection.

2.3.3.1 Evolutionary Algorithms

Darwin's evolutionary theory was the source of inspiration for the general population-based metaheuristic optimization algorithm known as EAs. EAs are a collection of heuristics for optimizing problems that simulate natural selection. EAs integrate mechanisms obtained from biological evolution, such as reproduction, mutation, recombination, and selection. The most prevalent type of EAs is GA. Implementing operations like recombination and mutation, one attempts to solve a problem in the form of strings of numbers (Chaudhry & Khan, 2016). Where the problem's candidate solutions are represented by the initial population, and the environment is the problem's solution space. The problem's best solution was selected after the candidate with the best fitness prevailed in a process of natural selection (Ezugwu et al., 2021). Other algorithms in this class of EAs include differential evolution, genetic programming, and evolutionary strategy.

A study on machine scheduling using EA was conducted by Durmaz et al. (2021), who proposed a GA to obtain near-optimum solutions to the tri-criteria single-machine scheduler. In addition, a fuzzy measure is proposed as an efficient tool to aggregate the criteria to reduce total weighted tardiness, maximum tardiness, and maximum earliness.

Meanwhile, another study of hybrid EA with other metaheuristic algorithms, i.e., Vela et al. (2020), proposed a hybrid evolutionary TS method with fuzzy sets modeling to optimize due-date satisfaction for JSSPs. At the same time, Golneshini and Fazlollahtabar (2019) proposed a hybrid metaheuristic GA with PSO to solve the hybrid FSSP with unrelated machines. It aims to reduce multi-objective total completion time and maximum lateness of jobs with fuzzy numbers for processing times and due dates.

2.3.3.2 Swarm Intelligence Algorithms

Most metaheuristic algorithms were created by observing nature. Optimizations based on the swarm intelligence demonstrated by fireflies, bats, and ants are just a few examples of how nature has attractive methodologies for solving computational problems (Hussain et al., 2019). The term "swarm" describes a group of dispersed moving creatures, such as insects, birds, or fish. Formally, a swarm is a group of homogeneous agents or individuals (Bansal & Pal, 2019). Artificial life, in general, and swarm theory, in particular, are connected to the primary algorithms of swarm intelligence, such as those used by fish schooling, fruit fly populations, bird flocks, bees, and ant colonies, and many more (Amira et al., 2018). Notably, scheduling is everywhere, in many real-world manufacturing environments, business planning, and holiday planning, to name a few.

Algorithms inspired by swarm intelligence offer many fundamental inherent advantages in the process of solving issues on a large scale (Chen et al., 2018). Although reasonable solutions to a complex optimization problem can be discovered using heuristics, there is no promise that the optimal results can be discovered. However, the use of SIAs has demonstrated promising results in solving the PMSPs. Additionally, several studies have been proposed to solve different multi-objective UPMSPs in fuzzy environments using different SIAs. Due to the fast-evolving worldwide technology, different industries currently confront more difficult scheduling and sequencing problems. Intelligent scheduling includes various tools and algorithms for effectively and successfully resolving scheduling problems. One significant class of optimization algorithms is SIAs.

Based on a review of SIAs for solving optimization problems by Tang et al. (2021), the nature-inspired heuristic algorithms are divided into four classes. That is, algorithms with biological, physical, evolutionary, and human behavior bases.

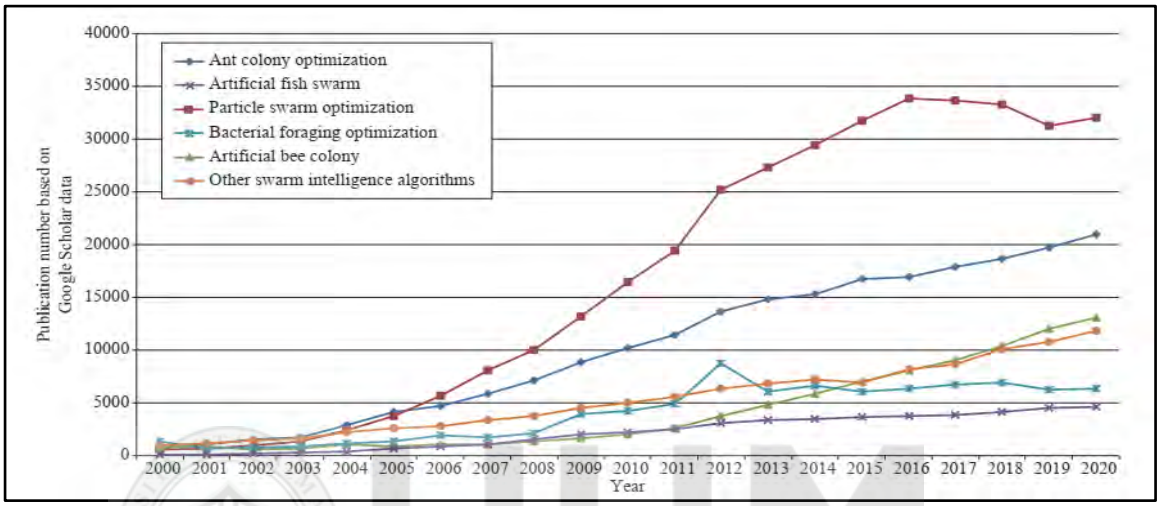


Figure 2.1. An indicator of swarm intelligence algorithm trends from 2000 to 2020 (Tang et al., 2021)

As illustrated in Figure 2.1, ABC, ACO, and PSO are the most used swarm algorithms in the scheduling problem, and all of them fall under biology-based algorithms. Many other less popular SIAs include FA, Artificial Fish Swarm Optimization (AFSA), and the Fruit Fly Optimization Algorithm (FOA) (Tang et al., 2021). Most swarm intelligence algorithms were first presented for continuous issues. However, some have also been developed for discrete optimization, with applications in various disciplines (Krause et al., 2013).

Pulansari (2021), for instance, provided a planned strategy for UPMSP utilizing the ACO algorithm to reduce the makespan and tardiness and compared the ACO algorithm with five heuristic algorithms based on the conditions and objectives. The findings suggest that

ACO is useful for inherent parallelism concerns and can give quick and acceptable results. On top of that, AFSA has recently been employed and developed in machine scheduling optimization problems. However, Lei et al. (2021) used an enhanced ABC algorithm to resolve a multi-objective distributed UPMSP to reduce total tardiness and makespan. Experimental outcomes indicate that the algorithm has effective and competitive performance in a deterministic environment. When contrasted with competitor intelligent algorithms,

2.3.3.3 Hybridization of Swarm Intelligence Algorithms for Solving Machine Scheduling Problems

In SIAs, the hybrid algorithms has become a crucial tool for raising performance. Combining two algorithms to increase their benefits while minimizing their drawbacks is known as hybridization. According to research, not all applications and functions are appropriate for every nature-inspired algorithm. Additionally, it does not offer the best exploration (search space exploration for promising solutions) and exploitation (exploiting the best solutions discovered so far) for each function during the search process. The power of the newly hybrid algorithms comes from their capacity to optimally combine exploration and exploitation (Singh et al., 2020). Many hybrid methodologies have been studied in scheduling problems. Kılıç and Organ (2025) proposed a new metaheuristic algorithm that uses variable neighbourhood search and local search selection to improve solution quality for UPMSP with setup times and minimising makespan. The algorithm was evaluated on a benchmark dataset and achieved more successful results. Based on the concept of TS, Chi et al. (2022) improved the Iterative Greedy (IG) algorithm and introduced the IG-TS algorithm for addressing UPMSPs with

the objective of minimizing makespan. The study compares three widely utilized hybrid algorithms in scheduling: the GA-TS algorithm, the ABC-TS algorithm, and the PSO-TS algorithm. The results indicate that the scheduling decisions derived from the IG-TS algorithm are the most effective and dependable. Al-qaness et al. (2021) suggested an algorithm which uses the advantages of two metaheuristics, the Whale Optimization Algorithm (WOA) and the FA, to minimize the makespan for UPMSP with sequence-dependent setup times. According to the results, the proposed method is better than eight common metaheuristic algorithms.

To improve the Salp Swarm Algorithm's (SSA) performance, Ewees et al. (2021) proposed a hybrid algorithm which uses the FA operators to improve the SSA's exploitation efficiency. Additionally, it works as a local search strategy to reduce the makespan, enhance searchability for UPMSPs, and boost solution quality. According to detailed comparisons, the proposed algorithm outperforms existing metaheuristics algorithms. A hybrid GA with a variable neighborhood structure algorithm was presented by Meng et al. (2020) to address the multi-objective problems of total tardiness and total makespan for identical PMSP and UPMSPs. The proposed algorithm, in comparison, outperforms the NSGAs II and III practically. Moreover, Ding et al. (2019) proposed a memetic algorithm to makespan minimization that blends an evolutionary UPMSP approach with an effective local search algorithms. The proposed hybrid algorithm offers a promising way to generate optimum or near-optimum results in real-time while maintaining high creativity, low cost, and rapid response to customer requirements.

2.3.3.4 Algorithms for Solving Unrelated Parallel Machine Scheduling Problems

Due to the complexity of the problem and the diverse optimization objectives and algorithms considered, various algorithms can be used to solve UPMSP. This subsection will discuss identifying and evaluating the current algorithms used for solving UPMSP. Based on a comprehensive literature review by Đurasević and Jakobović (2021), a vast variety of solution algorithms was proposed. It ranges from simple heuristics to complex hybrid metaheuristics that integrate multiple algorithms into one. The MILP model, a ϵ-constraint method, and the Nondominated Sorting GA (NSGA-II) are some of the efficient solutions offered by Yaghtin and Javid (2025) to address the complex multi-objective UPMSP with real-world constraints, like sequence-dependent setup times and periodic machine maintenance, in order to simultaneously minimize total tardiness, earliness, and total completion times. Similarly, Geurtsen et al. (2023) develop a MIP model to reduce makespan, total production completion times, and total job tardiness using machine eligibility constraints and setup times in UPMSP. To solve large instances at the industry scale, a hybrid GA with a novel solution representation is proposed. Tests conducted with real-world data demonstrate that the proposed model produces schedules that significantly enhance current factory practices. Furthermore, Safarzadeh and Niaki (2023) addressed a UPMSP using an exact method based on MP and ϵ-constraint approach to optimize multi-objective makespan and the total cost. The experimental results confirm the use of the MP solution method. Similarly, Zheng et al. (2022) developed a MILP model to produce optimal solutions for small-sized problems by minimizing the weighted sum of the total completion time of orders and the total processing cost of machines UPMSP. Two heuristic algorithms, type-based greedy and improved differential evolution, have been developed. The results of computational experiments demonstrate how efficient and

effective the proposed heuristic is in applications on a large scale. Two metaheuristic algorithms, SA and AIS, were considered for the UPMSP investigated by Shafipour et al. (2022), aiming to reduce the total early and late costs under varying constraints.

The outcomes reveal that SA utilizing the repair algorithm performs better than the other suggested algorithms. Åblad et al. (2021) developed and evaluated an exact solution method using the B&B algorithm of the UPMSP to reduce the makespan objective. The proposed algorithm is competitive when compared with various local search heuristics and MILP formulations on various instance types.

Fanjul-Peyro et al. (2019) proposed a new MILP and a new exact algorithm that minimizes setup times and makespan by combining a new procedure and some improvements with the suggested MILP models of the UPMSP. The results illustrate that the suggested algorithm significantly improves on existing algorithms and can solve extremely complex problems. In order to minimize the makespan on the UPMSP with sequence-dependent setup times, Arnaout (2020) developed the Worm Optimization (WO) algorithm.

The results of evaluations comparing the WO to other algorithms (TS, ACO, GA, SA) showed that WO was superior. In contrast, Jouhari et al. (2020) developed a new hybrid metaheuristic for solving UPMSPs that combines Harris Hawks Optimizer (HHO) and SSA to minimize the makespan. In terms of convergence to the optimal solution, the proposed algorithm outperformed both SSA and HHO as well as other algorithms.

Exact algorithms for UPMSP seemed ineffective for solving real-world instances of the problem until recently, and the proposed exact algorithms were only valid for relatively small to medium instances due to the complexity of UPMSP. Heuristic and metaheuristic algorithms have proven to be superior to exact approaches to solve large-scale instances

of the UPMSP. Due to the computational impracticality of solving with exact algorithms. The majority of the current literature focuses with heuristics and metaheuristics. However, they are easy to use, recreate, and adapt, improving our understanding of the issue. Hence, such algorithms have proven to be more applicable. Despite the abundance of studies on UPMSPs, it is expected that numerous designed solution methods involve hybrid and modified metaheuristic algorithms. To the best of our knowledge, no study has investigated the UPMSPs with the considerations described. As a result, the current study is an attempt to close this gap.

2.3.3.5 The Artificial Fish Swarm Algorithm

The AFSA is a continuous population-based algorithm that belongs to the biologically based swarm intelligence category. It was first introduced by Li (2002) and is inspired by the behaviors of a fish swarm. The algorithm demonstrated a significant ability to solve a range of NP-complete problems. This is due to the algorithm's scalability, fault tolerance, consistency, speed, flexibility, parallelism, and other qualities that make them suited for solving NP-complete problems (Pythaloka et al., 2015). This subsection reviews the AFSA literature and provides evidence from previous studies demonstrating that AFSA is a powerful and effective optimization algorithm for other complex problems. Recent studies (Alhaqbani et al., 2022; Pourpanah et al., 2023) show that the standard, modified, and hybrid models of AFSA have successfully addressed various real-world optimization problems. This involves data clustering, image segmentation, power allocation schemes, manufacturing, risk probability prediction, wireless sensor networks, and energy management. In scheduling problems, various researchers use and improve the AFSA. For instance, Pythaloka et al. (2015) used AFSA for JSSPs to generate an optimal solution containing the entire job's makespan. According to the search results, AFSA's efficiency

in JSSP was good. Meanwhile, Yadollahi and Razavi (2019) proposed utilizing an AFSA to present a new algorithm for addressing the university test timetabling scheduling problem and comparing this approach to existing algorithms to estimate the efficiency of the given method. The simulation results asserted that the suggested strategy outperformed other algorithms.

Nonetheless, the standard AFSA has some drawbacks despite its benefits. In particular, there is an imbalance between global and local search, slower convergence speeds later in the optimization process, a high time complexity, and a lack of benefit from group members' previous experiences in forecasting successive movements, among other issues (Alhaqbani et al., 2022). Accordingly, several attempts have been detailed in the literature to enhance and expand the capabilities of the standard AFSA by incorporating algorithms into AFSA as local search strategies to enhance the capacity of a particular search.

In Table 2.2, it was noted that the AFSA was improved with various algorithms to optimize single and multi-performance measures under different disciplines, including scheduling problems. To address the problem of reducing makespan in the steel industry, Sun et al. (2018) created a hybrid AFSA with Self-Adaptive Differential Evolution (SADE) for PMSP. The performance of the hybrid SADE-AFSA is improved regarding both efficiency and strength evaluations when comparing the suggested algorithm to the three other algorithms (AFSA, SADE, and PSO-DE). At the same time, Faeq et al. (2018) proposed a hybrid AFSA with Harmony Search (HS) for the Flexible Job-Shop Scheduling Problem (FJSSP) to reduce the makespan. Consider three sets of widely used benchmark instances to evaluate the performance of the AFSA-HS, demonstrating that the proposed AFSA-HS outperforms and is effective against several existing algorithms for FJSSP. In addition, Alobaidi and Hussein (2017) presented the hybrid AFSA with a

Variable Neighborhood Decent (AFSA-VND) strategy for the FJSP problem to minimize makespan, and benchmark instances are considered for performance evaluation. Compared to the standard AFSA, the experimental results suggest that AFSA-VND produces higher-quality solutions and converges more rapidly. Moreover, Ge et al. (2016) presented a hybrid AFSA with Estimation Distribution (AFSA-ED) for FJSSP to minimize makespan. The suggested algorithm has been assessed against well-known benchmark instances. The computations and comparisons indicate that the proposed AFSA-ED outperforms several existing algorithms. Additionally, Zhu and Jiang (2010) proposed that AFSA-TS performance combined with AFSA and TS would solve the problem with JSSP and reduce the makespan. A set of well-known JSSP benchmark problems of varying sizes is used to estimate the performance of the proposed algorithm. Simulated results reveal that AFSA-TS is an algorithm that can be used to solve the JSSP, and the outcomes are better than those from GA and AFSA. Conversely, Iranmanesh et al. (2017) developed a novel algorithm to considerably accelerate the convergence by hybridizing AFSA and group escaping behavior of fish for the traveling salesman problem to determine the shortest route by visiting each city only once and calculating the total distance. In essence, the suggested algorithm is more efficient, and the time taken to run it demonstrated that its convergence rate was significantly higher than that of the standard AFSA. The scheduling problem has been studied in the preceding literature with a single objective. In multi-objective, Tirkolaee et al. (2020) developed a hybrid method utilizing an interactive fuzzy solution algorithm and AFSA to minimize total costs and energy consumption in the FSSP. Additionally, other researchers used hybrid AFSA to tackle multiple objectives in different fields. For instance, Sharif et al. (2024) present a new model for software cost estimation problem for calculating the cost and time required for

software development based on a hybrid of AFSA and ABC algorithms. The combined method was evaluated using eight different data sets and eight different criteria. In the food enterprise management problem, Vitaliiovich and Leonidovna (2020) introduced hybrid algorithms based on AFSA and the grey wolf algorithm to solve multi-objectives by lowering costs and optimizing production, warehouses for finished goods, and raw materials.

On the other hand, Zuo et al. (2019) developed a hybrid AFSA with PSO for distributed artificial intelligence to address the problem of multi-agent cooperative work and path planning. Krishnaveni and Janita (2019), oppositional behavior learning added to AFSA for cloud computing to reduce cloud task scheduling execution cost and makespan. Moreover, Hajisalem and Babaie (2018) developed a hybrid algorithm based on ABC and AFSA for data mining and network behaviors to improve the speed and accuracy of intrusion detection systems.

Table 2.2
Hybrid AFSA's previous work in the literature

No	Authors	Hybrid AFSA	Problem	Objective	Limitations
1.	Sharif et al. (2024)	AFSA-ABC	Software Cost Estimation	Calculating the cost and time	This algorithm requires significant modifications to be directly applicable to scheduling problems
2.	Tirkolaee et al. (2020)	Hybrid algorithm of Self-Adaptive AFSA and interactive fuzzy solution algorithm	FSSP	Reduce total cost and total energy consumption	It not be designed for handling such complexity, especially with large-scale problems
3.	Vitaliiovich & Leonidovna (2020)	Hybrid AFSA and GWO	Food Enterprise Management Problem	Reduce costs and optimize production and warehouses for finished products and raw materials	This algorithm may have difficulty achieving an effective balance between conflicting objectives, leading to inefficient scheduling decisions.
4.	Zuo et al. (2019)	AFSA- PSO	Distributed Artificial Intelligence	Multi-agent cooperative work and path planning.	This hybrid algorithm fails to provide a extensive evaluation of the quality of the solutions obtained, and therefore there is

					no guarantee that it has provided optimal or even near-optimal solutions to the scheduling problem.
5.	Krishnaveni & Janita (2019)	Add oppositional behavior learning to AFSA	Cloud Computing	Reduce cloud task scheduling execution cost and makespan.	This hybrid algorithm is not well adapted to job scheduling problems because it is generally designed for static problems.
6.	Sun et al. (2018)	AFSA-SADE	PMSP	Minimize makespan.	The algorithm was developed for a specific problem in the special steel industry. Its efficacy may not extend to other types of scheduling problems.
7.	Faeq et al. (2018)	AFSA-HS	Flexible JSSP	Minimize makespan.	The algorithm might not handle trade-offs between multi-objectives well because it was primarily created for single-objective optimization.

8.	Hajisalem & Babaie (2018)	AFSA-ABC	Data Mining and Network Behaviors	Optimize intrusion detection system speed and accuracy.	The algorithm is effective for medium-sized datasets, but UPMSP instances can be combinatorially large, requiring more scalable and adaptive search techniques.
9.	Guo et al. (2018)	AFSA-GA	Remanufacturing of mechanical products	Find an optimal or near-optimal disassembly sequence.	Adapting this hybrid algorithm to a different problem may not fully leverage its strengths.
10.	AlObaidi & Hussein (2017)	AFSA-VND	Flexible JSSP	Minimize makespan.	This hybrid algorithm was designed to deal with specific FJSSP benchmark situations.
11.	Ge et al. (2016)	AFSA-ED	Flexible JSSP	Minimize makespan.	The main mechanisms of this algorithm are very specific to FJSP and may require the development of different exploitation strategies relevant to UPMSP.

12.	Iranmanesh et al. (2017)	Hybridize AFSA and group escaping behavior of fish	Traveling Salesman Problem	Determine the shortest route by visiting each city only once and calculating the total distance.	The hybrid AFSA that proved successful for TSP may not adapt directly or optimally to UPMSP.
13.	Zhu & Jiang (2010)	AFSA-TS	JSSP	Minimize makespan.	This algorithm uses an operation-based encoding that cannot represent the UPMSP. Also, Tabu memory causes a high memory and computational load for UPMSP with large-size instances.
14.	Jiang & Cheng (2010)	AFSA-SA	Clustering analysis	Solve some multimodal problems.	The algorithm was only assessed on a continuous benchmark function, so its scalability and robustness for large-size UPMSP are unproven.

Guo et al. (2018) developed a hybrid AFSA with a GA for mechanical product remanufacturing to obtain an optimum or near-optimum disassembly sequence. In clustering analysis, AFSA can be hybridized with SA to resolve multimodal problems (Jiang & Cheng, 2010).

According to past studies presented in Table 2.2, to boost AFSA's performance, many different algorithms have been put forth, whether combining AFSA with various algorithms or enhanced by adding behaviors to the algorithm. It is noted that Tirkolaee et al. (2020) is the only research that tackled hybrid AFSA for fuzzy scheduling problems to minimize multi-objective for FSSP. Moreover, AFSA has not previously been investigated as a modified or a hybrid algorithm for solving fuzzy multi-objective UPMSPs.

Generally, current AFSA literature shows that it has demonstrated strong performance in solving complex NP-hard optimization problems, but these studies also have specific weaknesses or limitations, such as poor scalability, inability to handle uncertainty, getting stuck in local optimization, or being designed to deal with a specific problem. This proven performance makes it a promising choice for a robust framework for developing new, modified, or hybrid algorithms. Therefore, this provides a clear research motivation for proposing a novel adaptation and demonstrating its effectiveness, thereby contributing a new and valuable solution for using the proposed fuzzy multi-objective UPMSP.

2.4 Standard Artificial Fish Swarm Algorithm

A continuous swarm intelligence-based optimization algorithm referred to as AFSA, developed by Li Xiao Lei (2002), involves random searching for a set of solutions to solve NP-hard problems (Pourpanah et al., 2023). Notably, AFSA has developed as an essential

tool in swarm intelligence optimization, characterized by its simple principle and notable attributes such as strength and acceptance for parameter settings (Peng et al., 2018). The fundamental idea is to mimic various ecological behaviors of fish by showing social search behaviors to relocate to areas with more consistent food availability. That is, a fish signifies a point or a fictitious entity of a true fish in a population, and the swarms move randomly (Villablanca, 2016). In essence, AFSA is an iterative algorithm that converges progressively to the optimal global solution (Cai, 2010). The main objective of this section is to provide a description of the AFSA, the basic procedure and standard behaviors of the algorithm that will be used as a framework for this research.

2.4.1 Description of Standard AFSA

In AFSA, candidate solutions are called Artificial Fish (AF). The best neighbor candidate solution depends on the AFs in the AFSA. The attractors of each AF are the best AF in its visual range and the swarm's center position. The AFSA considers every food position as a feasible solution to the optimization problem at hand, with the quality of the solution being measured by a fitness function based on the density of the associated food. The best solutions were posted on a bulletin board at the end of the process. A bulletin board is utilized by the AFSA to record the best position that has been identified thus far by all AFs. After each iteration, the algorithm compares the best AF (i.e., the AF with the highest fitness value) with the archived solution on the bulletin board. If the new AF is superior, the position on the bulletin board will be updated. The search cycle continues until the stopping criterion is satisfied (Pourpanah et al., 2023). Although maximum iterations are a typical stopping criterion for issues for which a solution is known, there is an additional stopping criterion method by determining the fitness value's variance. Note that if there is

minimal variance after several iterations, the procedure is supposed to be stopped (Cai, 2010). The most crucial parameters in AFSA include the visual distance ($visual$), which is the maximum Euclidean distance of vision for an AF. It controls how an AF views its surroundings and determines the closest neighbors for AF. The greater the visual value, the easier it is for the AF to determine the global extreme value and convergence (Pythaloka et al., 2015). Meanwhile, the step ($step$), representing the maximum step length that a fish can take at each movement, controls how an AF advances on the target. In addition, the crowding factor $\mu \in [0,1]$, $Trynumber$ is the number of attempts in preying behavior, and n a population sizes of AF are other parameters that affect AFSA performance (Almshhadany & Ibrahim, 2018). In the AFSA, the equilibrium among exploration and exploitation is controlled by $visual, step,$ and μ (Pourpanah et al., 2023). Accordingly, individual AF will choose different behaviors derived from their interpretation of environmental information. The AF in AFSA seeks the local solution within its visual scope, and the global optimal solution is determined through the cooperation of the fish swarm (Liu et al., 2019).

2.4.2 Descriptions of the AFSA Behaviors

Each AF employs four behaviors to identify positions with higher fitness values. These distinct behaviors of a fish swarm inspire AFSA and imitate decisions made by AF in the water, such as how they search for food and safety. The following is a description of AFSA's behaviors:

- **Preying behavior:** Fish have a basic behavior known as preying that allows them to move to locations with a higher concentration of food. This behavior is modeled within

a radius of a fish's neighborhood using the AF's current location and nearest neighbors as determined by the AF's field of view. Let X_i be the current position of the i^{th}AF, and X_j be an arbitrary state of an AF as follows:

$$X_j = X_i + Visual \times R\,(0,1), \tag{2.1}$$

where $Visual$ is the visual length between two AFs that are placed in X_i and X_j, the Euclidean distance can be defined as $dist_{ij} = \| X_i - X_j \|$, and R is a random vector with each element between 0 and 1. In minimization problems, if $f(X_j) \leq f(X_i)$, the i^{th} AF moves a step toward X_j using the following formula:

$$X_{prey} = X_i + \frac{X_j - X_i}{\| X_j - X_i \|} \times Step \times R\,(0,1), \tag{2.2}$$

where X_{prey} is the position of the i^{th} AF after executing the preying behavior, which is next to the $X_j - X_i$, and $\| X_{prey} - X_i \| \leq Step$.

However, if $f(X_j) > f(X_i)$, which would imply that the forward condition is not met, another position X_j is randomly selected by applying equation (2.1). If the fitness value, which could be the amount of food the AF is currently concentrating on, fails to be satisfied after trying for a given number of iterations, X_i randomizes its next step as follows:

$$X_{prey} = X_i + Visual \times R(0,1). \tag{2.3}$$

- **Swarming behavior:** In nature, fish prefer to stay close to the swarm to protect themselves from predators and avoid collisions within the group. In this behavior, the fish collect and move toward the center position to avoid crowding (Sun et al., 2019). Let n denotes the total number of AFs, n_f be the number of neighbors within the i^{th}

AF's visual range with $dist_{ij} < Visual$, and X_{center} be the central position of the swarm, which is the average of all AFs' positions:

$$X_{center} = \frac{1}{n}\sum_{n=1}^{k} X_k. \quad (2.4)$$

The i^{th} AF relocates in the direction of X_{center} if $f(X_{center}) \leq f(X_i)$ and $\frac{n_f}{n} \geqslant \mu$, where $\mu \in [0,1]$ is the crowding factor, indicates that there is greater food availability in the center and that the location is not overcrowded. As a result, X_i takes a movement toward the neighbor's center as follows:

$$X_{swarm} = X_i + \frac{X_{center} - X_i}{\| X_{center} - X_i \|} \times Step \times R(0,1). \quad (2.5)$$

Otherwise, the i^{th} AF will performs the preying behavior.

- **Following behavior:** In the moving process, a fish in a swarm discovers an area with a higher concentration of food and attracts the attention of their neighbors to quickly follow. For i^{th} AF, if at least another j^{th} AF is represented where $\| X_j - X_i \| \leq Visual$ and $f(X_j) > f(X_i)$ with a low crowded factor, i.e., $\frac{n_f}{n} < \mu$, then i^{th} AF moves a step toward the j^{th} AF position is as follows:

$$X_{Follow} = X_i + \frac{X_j - X_i}{\| X_j - X_i \|} \times Step \times R(0,1). \quad (2.6)$$

If there are no neighbors in the area around X_i or if all of them do not satisfy the condition, then i^{th} AF performs a preying behavior.

- **Random behavior:** This behavior is effective in eliminating the local optimum since each AF makes its own random search process and lets the fish swim freely or follow a swarm in a larger space. Furthermore, the calculation workload has been reduced, and algorithm efficiency has been improved (Cai, 2010).

The next state of AF is X_{next}, defined as follows:

$$X_{next} = X_i + V\text{isual} \times R(0,1). \quad (2.7)$$

These four behaviors are classified as fundamental by AFSA. However, researchers have attempted to implement additional behaviors to improve the algorithm's performance. This includes swallowing behavior (Jiang & Cheng, 2010), selection behavior (Huang & Chen, 2013), and escape behavior (Iranmanesh et al., 2017).

2.4.3 Procedure for the AFSA

AFSA starts by initializing n randomly chosen population sizes of AF and distributing them across the search space (Pourpanah et al., 2023). The next step is for each AF to simultaneously perform the swarming and following behaviors. In particular, AF moves in response to the environment and current conditions on subsequent behavior, and it impacts the environment over its own and other AF's activities. Once the requirements to execute the behaviors are not fulfilled, or when no improvement is observed after the behaviors have been performed, the AFSA executes a preying behavior. Figure 2.2 illustrates an outline of an AF and its surrounding environment. Where X_i and X_{next} designate the current and following positions of the i^{th} AF, respectively. Meanwhile, X_j is a position selected at random within the line of vision and $X_{n,1}$, $X_{n,2}$, and $X_{n,3}$ are AFs that fall within the i^{th} AF's visual range.

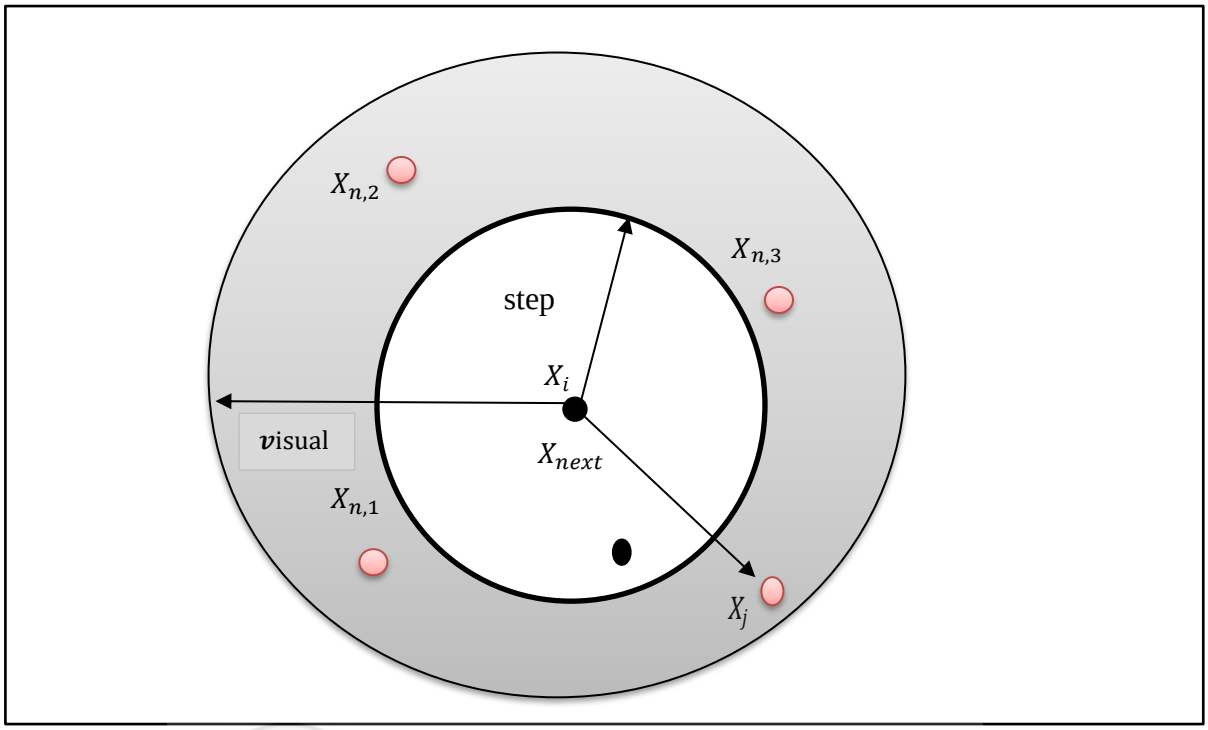


Figure 2.2. An artificial fish and its environment.

The AFSA procedure is summarized in Figure 2.3. Until the stop condition is satisfied, these procedures are repeated iteratively, with each fish modifying its position in accordance with behaviors and evaluations of fitness. In addition, by imitating the natural behavior of a fish swarm, the objective is to identify the best solution to the problem (Pourpanah et al., 2023).

```
Input: Initial parameters;
Output: Optimal solution;
Generate n random AFs within the search space;
while the stop condition is not met do
    for each AFs do
        Compute the fitness value;
        Execute the behaviors;
    end
    Update the bulletin board.
end
```

Figure 2.3. The AFSA procedure

Figure 2.4 displays the flowchart of the standard AFSA. The algorithm starts by randomly generating *n* AF in the search space, while each AF represents a solution candidate. For each AF, the fitness function determines its quality. It also helps determine better solutions. The movement of AF is influenced by current and environmental factors, which in turn impact the environment through its own actions and those of other AF. The best solutions are put on a bulletin board to guide the search. The process turns into two possible behaviors: following and swarming. If the movement conditions are met, perform the appropriate behavior. Conversely, if the conditions are not met, use the preying behavior to search for better solutions. Subsequently, each AF's position is updated based on the behavior performed. The best solutions are updated on the bulletin board to indicate progress. Finally, the algorithm determines whether the stopping condition is met. That involves the maximum number of iterations reached or the solution convergence achieved.

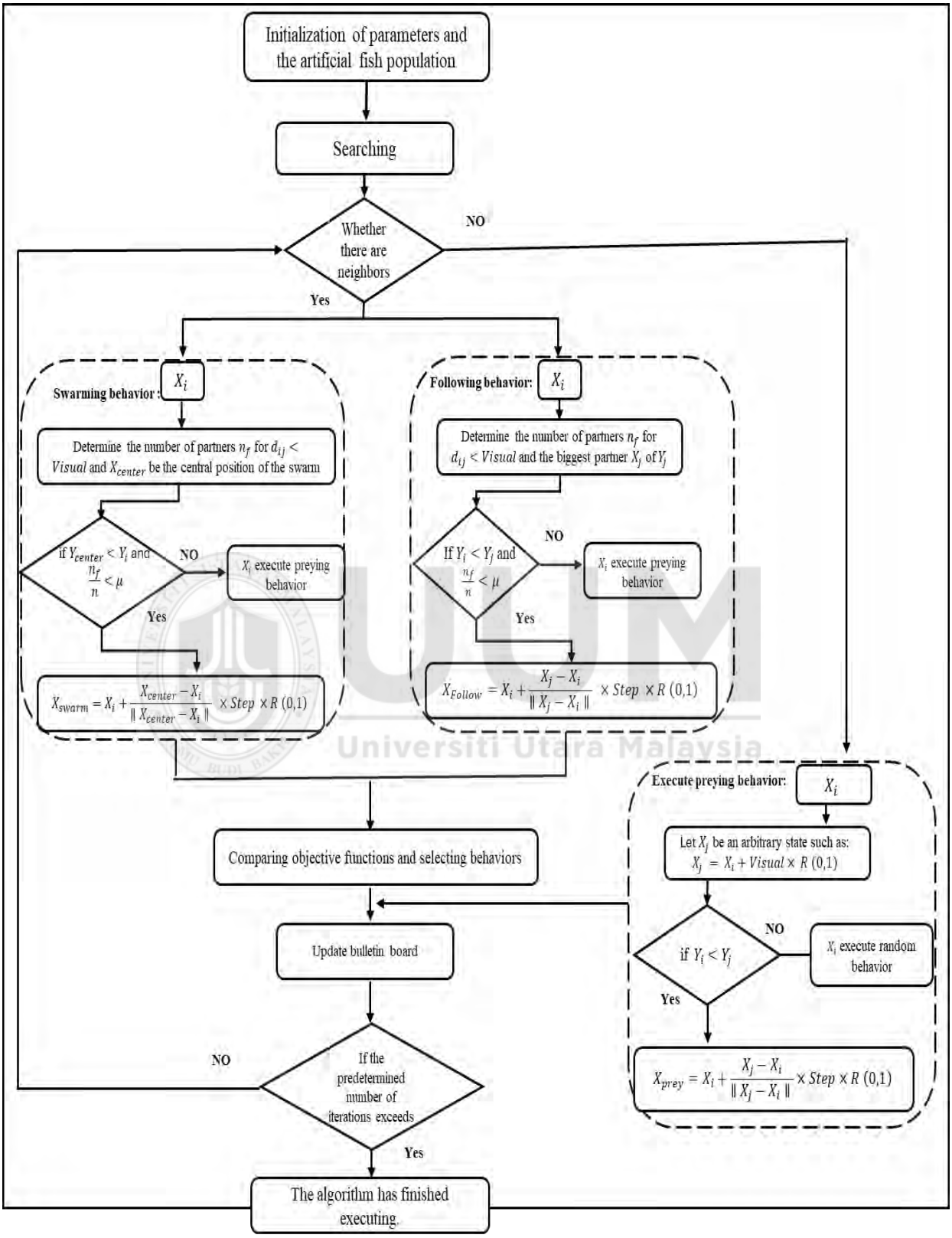


Figure 2.4. Flowchart of the standard AFSA

2.5 Modified Artificial Fish Swarm Algorithm for Solving Unrelated Parallel Machine Scheduling Problem

AFSA is well-suited for solving NP-hard problems due to its parallel computational capability and adaptive global search ability, making it applicable across various fields (Liu et al., 2019). However, it also has several drawbacks when dealing with multi-objective and nonlinear problems. These include inefficient convergence, susceptibility to local optima, and weak optimization performance. In recent years, scholars have proposed various attempts to improve AFSA's shortcomings. The primary motivation for reviewing modified AFSA for the UPMSP is to study the different modification algorithms that have been effectively employed and provide evidence from previous studies that modifying the standard AFSA is widely used for enhancing its efficiency and adapting it to this problem domain. The improved and modified AFSA makes it a valuable tool that can be efficiently applied in various applications. For example, Zhao et al. (2023) adopted improved AFSA for route planning of autonomous vessels by introducing a directional operator to improve efficiency, a probability weight factor to avoid local optima, and an adaptive operator to achieve better convergence performance. Meanwhile, Jin et al. (2023) suggested a modified AFSA for unit commitment optimization in power systems to deal with the original algorithm's drawbacks including premature convergence and local extremes. Variable vision, movement strategy adjustment, and combining the mutation operation of Genetic Algorithms (GA) are among the improvements. At the same time, Wang et al. (2022) suggested a modified AFSA combined with a local path optimizer to solve the path planning problem for unmanned surface vehicles. They tailored four operators and implemented an adaptive factor to improve the algorithm's convergence performance. In

addition, Huang et al. (2021) proposed three adaptive step methods to speed up and deal with the weaknesses of the standard AFSA for solving sensor layout optimization problems in fiber grating networks. Simultaneously, Li et al. (2021) proposed improving AFSA to find the best scheduling, which was formulated as a multidimensional knapsack problem to minimize the time delay. Added to that, Tan and Mohamad-Saleh (2020) developed a new algorithm based on AFSA to improve the performance of global and local search algorithms in optimization. The proposed algorithm includes several improvements, such as hybridizing characteristics, introducing normative communication and memory behaviors, and adapting parameters in terms of visuals and steps. In addition, Gao et al. (2020) introduced a novel AFSA using Cauchy mutation to quicken the algorithm's convergence rate to solve the parameter selection problem for the twin support vector machine. On that note, Sureja and Patel (2020) examined a novel algorithm generated by fish colony foraging social behavior, which is then applied to a combinatorial problem known as the random traveling salesman problem. In line with that, Liu et al. (2019) proposed a modified AFSA to solve water supply network problems by updating the equations of behaviors to make changes in the search process and executing mutation operations with the optimal individual. Furthermore, Krishnaveni and Janita (2019) suggested a modified AFSA that added oppositional behavior learning to standard AFSA to make a reverse population that improved task scheduling and significantly minimizing multi-objective makespan and execution costs. The proposed algorithm is more effective than the current techniques compared to Particle Swarm Optimization (PSO) and GA algorithms. Moreover, the modified AFSA proposed by Peng et al. (2018) based on a random searching strategy called Levy flight to enhance preying behavior and firefly behavior by incorporating the Firefly Algorithm (FA) moving strategy to control random

movement after AFSA has determined the direction. The proposed algorithm outperforms the test algorithms in terms of convergence speed and optimization accuracy on several benchmark problems. Additionally, these modifications attempted to improve the algorithm's performance in terms of convergence, optimality, and escape from local optimality, to name a few. However, until now, the algorithm has continued to be developed. While there is minimal research introducing modified AFSA in machine scheduling, Tirkolaee et al. (2020) introduced Self-Adaptive AFSA (SAAFSA) for solving Flow Shop Scheduling (FSS) problems to minimize total cost and total energy consumption. In particular, the SAAFSA is an improved version where self-adaptive behaviors are introduced to enhance the algorithm's performance.

To the best of our knowledge and based on this comprehensive review, no study has yet attempted to modify AFSA and use it to solve UPMSP. In addition, although many improvements and modifications to the fish swarm algorithm have been proposed regularly, they have primarily focused on balancing the processes of exploration and exploitation. However, there are no detailed dealings to provide a guide towards accurate and precise directions (Tan & Mohamad-Saleh, 2020).

2.6 Similar Study

In this section, the previous most closely linked published works are described to gain a better understanding of previous works in this research direction. Hybrid AFSA with interactive fuzzy solution method based on the ε-constraint method to simultaneously minimize the multi-objective total cost of the production system. Meanwhile, the total energy consumption in FSSP is studied by Tirkolaee et al. (2020), and the results signify the high performance of the proposed algorithm for problems of varying sizes.

Furthermore, Kongsri and Buddhakulsomsiri (2020) developed the MILP model to represent UPMSP with sequence-dependent set-up time to reduce makespan and total tardiness in a deterministic environment. However, because of the complexity of the UPMSP problem, the model is limited to solving small problem instances and requires lengthy computational tests and long solution times.

On the other hand, Sadati et al. (2019) used TrFNs for the primary parameters to investigate UPMSP for minimizing fuzzy multi-objective makespan and maximum tardiness. They used two exact algorithms (i.e., two-phase fuzzy and ε-constraint) to solve small-sized problems to obtain a set of Pareto solutions and two metaheuristics to solve medium and large-sized problems. At the same time, Sabti (2017) created two hybrid metaheuristic algorithms based on the Sequence Job Minimum Completion Time (SJMCT) algorithm and multi-objective hybrid EAs. The author named the first SJMCT algorithm with NSGA (SJMCT-NSGA-II) and the second SJMCT algorithm with Strength Pareto Evolutionary Algorithm (SJMCT-SPEA-II). They aim to minimize makespan and total tardiness for UPMSP in a deterministic environment, and the two hybrid algorithms are efficient and practical for solving large-scale problems by identifying Pareto optimal solutions. Furthermore, SJMCT-SPEA-II outperforms SJMCT-NSGA-II in terms of both efficiency and running time.

2.7 Chapter Summary

This chapter presents an in-depth review of the existing literature on fuzzy MSPs addressed using various solution methods under both single- and multi-objective

frameworks. Based on the reviewed studies, particular emphasis is placed on multi-objective MSP methodologies, as industrial practices typically require the simultaneous optimization of multiple, often conflicting, objectives. Nevertheless, identifying optimal job sequencing that satisfies the requirements of a large number of jobs and machines remains a challenging scientific problem. Moreover, only a limited number of studies have addressed fuzzy UPMSPs involving multiple objectives simultaneously.

Motivated by this research gap, and building upon the work of Kongsri and Buddhakulsomsiri (2020), this study proposes the development of a MILP model for the UPMSP that excludes sequence-dependent setup time constraints. The proposed model aims to simultaneously minimize makespan and total tardiness within a fuzzy environment, thereby enhancing model flexibility and applicability to real-world scheduling scenarios.

Although many optimization algorithms have demonstrated promising performance in solving scheduling problems, no single method consistently outperforms others across all problem instances. Consequently, the development and evaluation of new metaheuristic algorithms remain an active and challenging area of research. In this context, SIAs have emerged as effective tools for solving UPMSPs under uncertainty, due to their adaptability and strong global search capabilities. A comprehensive review of the literature reveals that, to date, no study has proposed the AFSA, either in modified or hybrid form, for solving fuzzy multi-objective UPMSPs where both processing times and due dates are represented by fuzzy numbers.

Given that AFSA is inherently a continuous optimization algorithm, while UPMSP is a discrete combinatorial problem, it is essential to modify the standard AFSA framework to

enhance its search capabilities and enable its application to discrete scheduling environments. Accordingly, this research adopts AFSA as the foundation for developing modified and hybrid algorithms, thereby motivating its use as a novel and effective algorithm for solving the proposed fuzzy multi-objective UPMSP.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction

This chapter outlines the research process and methodology used to achieve the research objective. The following section provides a detailed explanation of the five phases of the research process implemented in the methodology. Section 3.3 concludes the chapter with a summary.

3.2 Research Process

The research process in Figure 3.1 represents the structured approach adopted to achieve the research objectives. Each phase in the figure reflects a critical component of the research methodology, from identifying the research problem to evaluation. By illustrating the systematic flow of activities, the figure serves as a roadmap, ensuring clarity and coherence throughout the research process. The following section discusses the five phases outlined in Figure 3.1, explaining each phase and its role in the overall research process.

3.2.1 Phase 1: Problem Definition

This phase is comprehensively covered in Chapters 1 and 2 of this work. Chapter 1 begins with the background of Machine Scheduling Problems (MSPs) and the research objectives that must be solved, providing direction and motivation for this study. This study's significance and scope have been defined to develop a new modified and hybrid algorithm for solving the fuzzy multi-objective Unrelated Parallel Machine Scheduling Problem (UPMSP). Furthermore, Chapter 1 also introduces the critical foundation for all concepts related to MSPs.

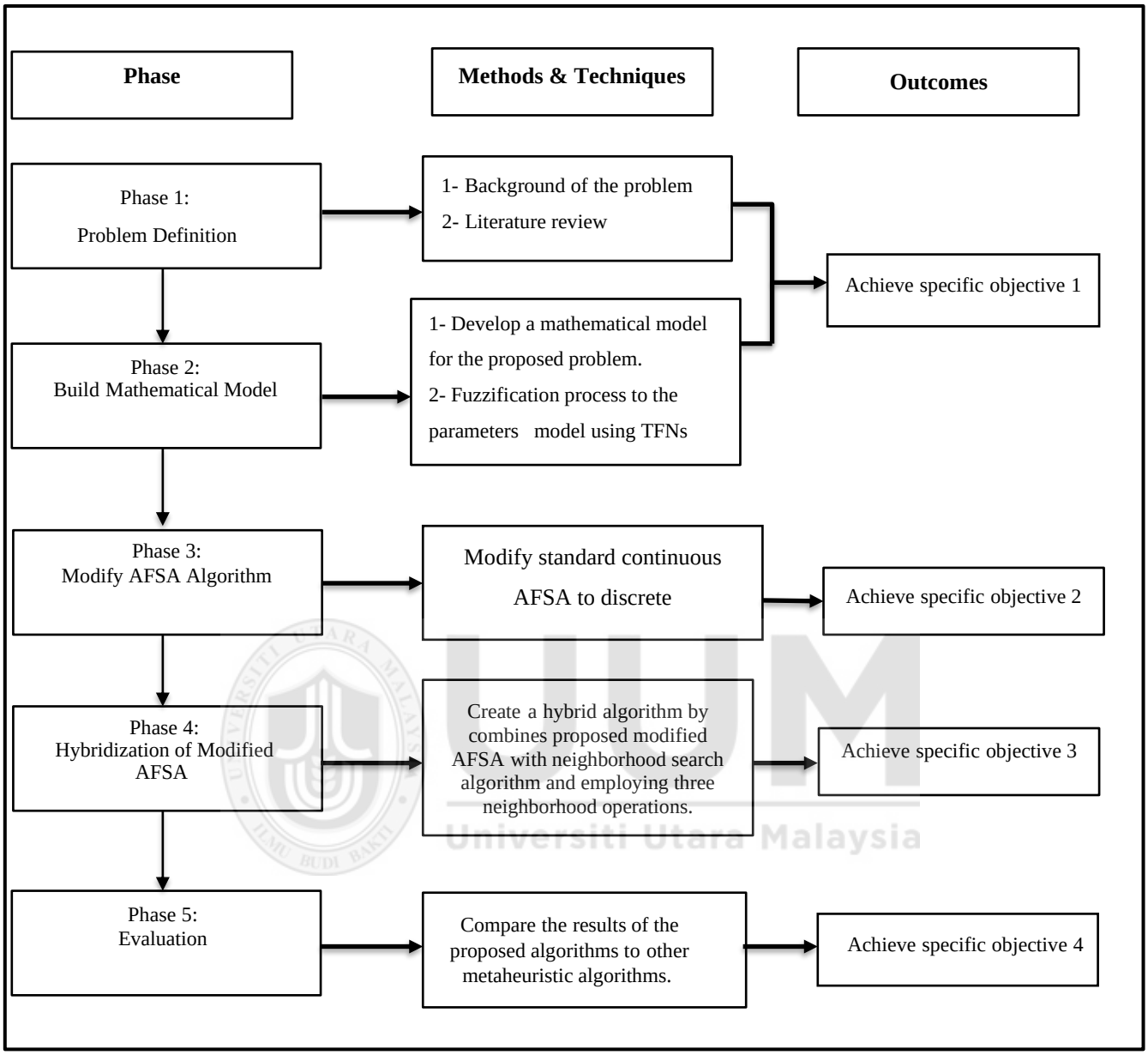


Figure 3.1. Details of the research process.

Following that, Chapter 2 summarizes relevant research, examining existing studies on deterministic and fuzzy multi-objective scheduling and solution algorithms across four MSPs. It concludes with an analysis of the advantages and disadvantages of previously employed algorithms for solving multi-objective models, highlighting the significance of this work.

Furthermore, a number of related works of literature discuss the use of various algorithms to solve scheduling problems for fuzzy machines. This continues by presenting the standard artificial fish swarm algorithm (AFSA) and modified AFSA for solving UPMSP. The aim of MSPs is to best assign limited resources to processing jobs within specific timeframes, ensuring efficiency and effectiveness. To address the diverse requirements of real-world applications, various constraints have been considered when designing scheduling solutions. Table 3.1 highlights important classes of MSPs, along with numerous additional constraints and common objective functions (Çolak & Keskin, 2022; Đurasević & Jakobović, 2021).

Table 3.1

Examples of machine environment, constraints, and objective functions

Machine environment (α)	**Constraints (β)**	**Objective functions (γ)**
1 Single machine	rj Release dates	C_j
P_m Identical parallel machine	$Prmp$ Pre-emptions	$Cmax$
Q_m Uniform parallel machine	$Prec$ Precedence	$Lmax$
R_m Unrelated parallel machine	$Prmu$ Permutation	$\sum Tj$
F_m Flow shop	$sijk$ setup times	$\sum wjCj$
J_m Job shop	$batch$ Batch processing	$\sum Uj$
O_m open shop	$Brkdwn$ Breakdowns	

In the second column of Table 3.1, relevant constraints are listed and briefly summarized. The constraints are as follows:

- Release dates (rj): Also referred to as the ready date, they display the earliest date when a job could be managed by a machine.

- Preemptions ($Prmp$): It indicates that a job does not have to be run on a machine from beginning to end.
- Precedence ($Prec$): It signifies how different jobs are ranked in order of importance.
- Permutation ($Prmu$): It states that every production stage processes all jobs in the same order.
- Setup times ($sijk$): A specific setup operation must be conducted on a machine when changing from one job to another in order to prepare it for the new job. The operation's time is indicated by the notation $sijk$. In particular, i stands for the job that has finished running, j stands for the job that will be performed next, and k refers to the machine that processes the jobs.
- Batch processing (batch): It indicates that jobs should be processed in batches rather than individually, with each machine capable of overseeing a group of jobs at once.
- Breakdowns (Brkdwn): It suggests that a machine's breakdown could result from a stochastic event, making it unavailable.

The scheduling optimization problem has been included to optimize a single objective. However, decision-makers often aim to simultaneously optimize several different objectives, leading to Multi-Objective Optimization (MOO). Note that a solution in MOO may not be the optimal solution for all objective functions, making it challenging to define what constitutes an optimal solution (Deb, 2014).

The objective functions listed in Table 3.1 were studied previously, and various combinations of these objectives are considered multi-objective. In addition, job completion times have a direct impact on scheduling performance objectives such as:

- Completion time (C_j): specifies when job j ends in the system.
- Makespan (C_{max}): the maximum completion time of schedule X of N jobs.
- Maximum Lateness (L_{max}): it is the worst possible violation of due dates (d_j), lateness (L_j) is the duration by which a job's completion time exceeds its (d_j).
- Total Tardiness ($\sum T_j$): tardiness $T_j = max\ (C_j - d_j, 0) = max\ (L_j, 0)$ is the consequence of a job's positive lateness (Munoz-Estrada, 2020).
- Total Weighted Completion Times ($\sum W_j C_j$): minimization of this objective shows a high level of client satisfaction due to job completion on time, the weight w_j of job j is essentially a primary factor indicating the significance of job j in comparison to other jobs in the system (Pinedo, 2008).
- The number of tardy jobs ($\sum U_j$): this suggests how many late jobs or unfulfilled customer orders there are; U_j is one if $C_j > d_j$; 0 otherwise (Molaee & Moslehi, 2014).

As a result, job completion times are critical for schedule performance, and one of the most extensively researched scheduling objectives is the makespan (Yepes-Borrero et al., 2020). Despite the common usage of deterministic parameters such as time parameters and constraint conditions, other relevant information can be used more as uncertainty parameters to deal with scheduling uncertainties, produce more realistic results, and make the MSPs more flexible and practical.

The MSP's process is fraught with uncertainties. Model parameters and environmental data, such as due dates and processing times, are very frequently occurring sources of uncertainty due to imprecision and vagueness resulting from personal bias and subjective opinion. The uncertainty of scheduling must, therefore, be considered to improve the

schedule's quality and make it more reflective of real-world scheduling issues (Floudas & Lin, 2004).

The fuzzy set theory is more advantageous and computationally simpler for representing and explaining uncertainties compared to probability theory when there is a lack of data, particularly when many variables are involved (Liao & Su, 2017). A membership function describes a fuzzy number, allowing it to solve large-scale problems (Salimifard et al., 2019). In addition, fuzzy set theory allows for the possibility of creating new approaches to scheduling issues that are more realistic and reliant on the user's knowledge and experience when dealing with uncertain data.

Phase 1 emphasizes the significance of makespan and total tardiness as multi-objective function in the UPMSP environment, as well as the use of fuzzy numbers for main time parameters, through an explanation of the problem and a literature review. Then, next phase involves developing a mathematical model for the proposed problem and applying fuzzy set theory to the parameters.

3.2.2 Phase 2: Build Mathematical Model

This phase deals with scheduling non-identical jobs $J_1, J_2, \dots, J_n$ on the unrelated parallel machine $M_1, M_2, \dots, M_m$, where every job j has a processing time p_{ij} and a due date d_j. The proposed multi-objective UPMSP can be indicated as $R//(C_{max} + \sum T_{ij})$ (Graham et al. 1979), where C_{max} signifies the time when the last job leaves the system and T_{ij} is a popular performance measure where it is constantly preferred to conclude an earlier task than a later one. Meanwhile, the amount of time past the deadline is referred to as job tardiness. In line with this, one of the most crucial requirements in many manufacturing systems is minimizing total tardiness, particularly in the current environment when

competition is intensifying (Chaudhry & Elbadawi, 2017). Makespan and total tardiness minimisation are the most studied as a multi-objective UPMSP (Meng et al., 2020).

In 2020, Kongsri and Buddhakulsomsiri formulated a Mixed Integer Linear Programming (MILP) model to minimize a combination of both makespan and total tardiness with sequence-dependent Setup times for UPMSP.

Building on the problem definition from Phase 1, the aim of Phase 2 is to revise the mathematical model developed by Kongsri and Buddhakulsomsiri (2020) by removing the sequence-dependent setup time constraint. The sequence-dependent setup time constraint was removed since it requires high memory requirements and lengthy solution times, as verified by computational tests in the (Kongsri & Buddhakulsomsiri, 2020) model. As the complexity of UPMSP with significant instances of many jobs and machines and varying processing times for each job increases due to the modeling of multiple objectives in NP-hard problems with fuzzy parameters, it is vital to simplify the mathematical models by reducing constraints to make them more flexible and expandable. This makes it more general and easier to apply the fuzzification process by fuzzifying the mathematical model with suitable fuzzy membership functions for the main parameters, processing time, due date, and objective functions.

- **Problem Description and Mathematical Model**

The proposed problem description contains two objectives, where the makespan and total tardiness are formulated to be reduced at once. First, the multi-objective UPMSP model is created with crisp values for the time parameters (processing time and due date) as presented in equations (3.1) and (3.2):

$$C_{max} = max\, C_{ij}\,, \qquad \forall i\,, j \quad i = 1, \dots, m, j = 1, \dots n, \tag{3.1}$$

where C_{ij} is the completion time of job j on machine i.

$$T_{ij} = max\big(0, C_{ij} - d_{ij}\big) \quad \forall i, \text{j}, \tag{3.2}$$

where T_{ij} is the tardiness of job j *on* machine i, if the completion time C_{ij} of job j on machine i is greater than its due date $d_{ij,}$ then this job is considered tardy. Otherwise, the tardiness of job j is equal to 0.

As stated above, the two objectives are reduced at once. The following is the suggested multi-objective mathematical model for UPMSP, where:

Assumptions: The subsequent presumptions are taken into account prior to problem formulation:

1- All jobs and machines are available at time zero
2- Parallel machines are unrelated (the processing time of each job on each machine differs and depends on the machine).
3- Preemption is not allowed.
4- Processing times and due dates are all deterministic input parameters.
5- Only one job can be processed by each machine at a time.

Indices and sets:

N: the total number of jobs.

M: the total number of machines.

j, k: index for jobs, $j, k \in N = \{1, \dots, n\}$, N_0 denotes the set of jobs, including a dummy job, $N_0 = \{0,1, \dots, n\}$.

i: index for machine, $i = 1, \dots, M$.

Parameters and decision variables:

- p_{ij} : The processing time of job j on machine i.
- d_j : The due date of job j.
- C_{ij} : The completion time of job j on machine i.
- C_{max} : Maximum completion time of the schedule (makespan).
- w = The weight value.
- v : The sufficiently large number.
- T_{ij} : The tardiness of job j on machine i.
- $X_{ijk} = \begin{cases} 1, & \text{if job } j \text{ immediately follows job k in sequence on machine } i \\ 0, & \text{Otherwise} \end{cases}$.

The original model of Kongsri and Buddhakulsomsiri, (2020), is stated here:

$$min\ \alpha(C_{max}) + (1-\alpha)\sum_{j \in N} e_j^+ , \forall j \in N \tag{3.3}$$

Subject to

$$\sum_{i \in M} \sum_{j \in N_0 : j \neq k} X_{ijk} = 1 \ , \forall k \in N \tag{3.4}$$

$$\sum_{i \in M} \sum_{k \in N_0 : j \neq k} X_{ijk} = 1 \quad , \forall j \in N \tag{3.5}$$

$$\sum_{k \in N_0 : j \neq k} X_{ijk} - \sum_{h \in N_0 : j \neq k} X_{ihj} = 0 \quad \forall\, i \in M, \forall j \in N \tag{3.6}$$

$$\sum_{k \in N} X_{i0k} \leq 1 \quad \forall\, i \in M \tag{3.7}$$

$$C_k - C_j + v\left(1 - X_{ijk}\right) \geq S_{ijk} + p_{ik} \quad , \forall\, i \in M, \forall\, j \in N_0, \forall k \in N : j \neq k \tag{3.8}$$

$$C_0 = 0 \tag{3.9}$$

$$C_j \leq C_{max} \quad \forall j \in N \tag{3.10}$$

$$C_j - d_j = \quad e_j^+ - e_j^- \qquad \forall j \in N \tag{3.11}$$

$$X_{ijk} \in \{0,1\} \, , \forall\, i \in M, \forall\, j \in N_0, , \forall\, k \in N_0 : j \neq k \tag{3.12}$$

$$C_j, \;\; e_j^+, e_j^- \geq 0 \, , \forall j \in N \tag{3.13}$$

The equation (3.3) represent objective function of minimize the makespan and total tardiness. Constraints (3.4) force that each job has exactly one preceding job. Constraints (3.5) indicate that each job has exactly one succeeding job. Constraint (3.6) represent flow balance constraint of each job at each machine. Constraint (3.7) specify the first job on each machine. Constraint (3.8) ensure the completion time of job *j* is equal to completion time of the preceding job plus setup time between these two jobs and processing time of job *j* at each machine. Constraints (3.9) sets the completion time of dummy job to 0. Constraints (3.10) force the completion time of each job *j* is not greater than the makespan. Constraint (3.11) that compute the tardiness value using variable e_j^+ (tardiness) and e_j^- (earliness) and constraints (3.12)-(3.13) specify type of decision variables.

After adopting the linear multi-objective model presented by Kongsri and Buddhakulsomsiri (2020) and removing the time constraint, the proposed multi-objective model for UPMSP is a revised model, which is formulated as follows:

$$Min\ F\ = wC_{max} + (1-w)\sum_{j=1}^{n}\sum_{i=1}^{m} T_{ij}. \tag{3.14}$$

Subject to:

$$\sum_{i\in M}\sum_{k\in N_0, k\neq j} X_{ijk} = 1 \qquad \forall\, j \in N, \tag{3.15}$$

$$\sum_{i\in M}\sum_{j\in N_0, j\neq k} X_{ijk} = 1 \qquad \forall\, k \in N, \tag{3.16}$$

$$\sum_{j\in N_0, j\neq k} X_{ijk} - \sum_{h\in N_0, j\neq k} X_{ihk} = 0 \quad \forall\, i \in M, k \in N, \tag{3.17}$$

$$\sum\nolimits_{j\in N} X_{i0j} \leq 1 \quad \forall i \in M, \tag{3.18}$$

$$C_0 = 0, \tag{3.19}$$

$$C_{ij} - C_{ik} + v\left(1 - X_{ijk}\right) \geq p_{ik} \quad \forall i \in M, \forall k \in N_0, \forall j \in N: j \neq k, \tag{3.20}$$

$$C_{max} \geq C_{ij} \quad \forall j \ \in N, \tag{3.21}$$

$$T_{ij} \geq C_{ij} - d_j \quad \forall j \in N, \tag{3.22}$$

$$p_{ij}, C_{ij}, T_{ij}, C_{max} \ \geq 0, \tag{3.23}$$

$$X_{ijk} \in \{0,1\}. \tag{3.24}$$

Equation (3.14) signifies the multi-objective functions to minimize the makespan and total tardiness. Constraint (3.15) guarantees that every job is executed by only one machine. When job j^{th} is performed after the job k^{th} on the machine i^{th}, the X_{ijk} value is equal

to 1. Otherwise, the value will be 0. Meanwhile, Constraint (3.16) demonstrates that there is only one job that is scheduled first at each machine, which implies $X_{ijk} = 1$ when the k^{th} job is the first job on a machine i^{th}, otherwise $X_{ikj} = 0$. Consequently, Constraint (3.17) specifies the job's flow balance at each machine, which indicates only one preceding job and one succeeding job. Furthermore, Constraint (3.18) specifies the machine's initial job. Constraint (3.19) gives zero completion time for the dummy job. Constraint (3.20) confirms that the completion time of the job j^{th} equals the completion time of the preceding job plus the processing time of the job j^{th}at each machine. This can be executed using a large number v. When $X_{ijk} = 1$, and if the job j^{th} is ordered after the job k^{th}, then $v(1 - X_{ijk}) = 0$ and $C_{ij} = C_{ik} + p_{ij}$. Otherwise, when job j^{th} is not ordered after the job k^{th}, then $X_{ijk} = 0$, and thus, $v(1 - X_{ijk}) = v$. Constraint (3.21) shows that the maximum completion time for all machines is equal to the makespan, while Constraint (3.22) calculates the value of each job's tardiness. Constraint (3.23) establishes that none of the decision variables can take on a negative value, and Constraint (3.24) explains the binary variables.

3.2.2.1 Fuzzification Model

A fuzzy scheduling problem is a classic scheduling problem with some imprecision in variable coefficients, constraints, or even objectives. In contrast, the main goal of a fuzzy scheduling process is to obtain the "best" solution (decision) in the presence of uncertain data. Notably, fuzzification is the procedure of transforming a crisp set into an appropriate fuzzy set (Chakraverty et al., 2019).

- **Fuzzy Set Theory**

Zadeh (1965) presented the theory of fuzzy sets as a strong and valuable instrument that could be used in a variety of situations. This includes operations research, mathematical modeling, control theory, and a wide range of practical applications (Nasseri & Ebrahimnejad, 2011). Fuzzy set theory offers a convenient framework that analyzes production scheduling processes mathematically. Fuzzy set are also ideally suited as a strategy for complicated systems in which the optimization parameters are interpretations of fuzzy ideas (Yeh et al., 2014). In other words, fuzzy set theory satisfy our desire to convert human knowledge into mathematical representations.

- **Fuzzy Set**

A fuzzy set extends the traditional framework of classical set theory, broadening the integer range of 0–1 to encompass the continuous values within [0,1], where a fuzzy set A is described throughout a set of all real numbers *R* by a function of membership μ_A:$R \longrightarrow [0,1]$. A fuzzy set A of *R* is described by a membership function $\mu_A(x)$, which a real number within the interval [0, 1] is related to each element *x* in *R*. The function $\mu_A(x)$ denotes the grade of membership of x in *R*. The larger $\mu_A(x)$, the stronger the degree of belonging for *x* in *R* (Zadeh, 1976).

Definition 3.1 (Zimmermann, 2010): A fuzzy set $\tilde{A}$ in X is defined by the function $\mu_{\tilde{A}}: X \longrightarrow [0,1]$ is a set of ordered pairs if X is a collection of items denoted by x:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)): x \in X\}. \quad (3.25)$$

As displayed in Figure 3.2, the membership function $\mu_{\tilde{A}}(x)$ of the fuzzy set $\tilde{A}$ is the relationship between various crisp values *x* and values in the interval [0,1]. The

membership function's range is a finite subset of non-negative real numbers. The degree of object membership x to the main set X is described by $\mu_{\tilde{A}}(x)$. Note that x is not included in the fuzzy set A if $\mu_{\tilde{A}}(x) = 0$, fully included in the fuzzy set A if $\mu_{\tilde{A}}(x) = 1$, and partially included in the fuzzy set A if $0 < \mu_{\tilde{A}}(x) < 1$.

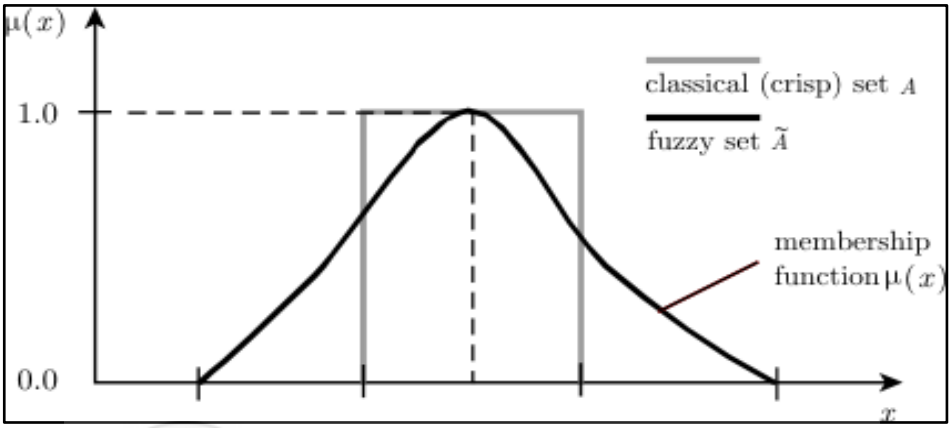


Figure 3.2. Crisp set A with fuzzy set $\tilde{A}$

Definition 3.2 (Gani & Assarudeen, 2012): The elements of the universe X whose membership values surpass the threshold level α are included in the fuzzy set $\tilde{A}$ is α - cut or α - level set, named as $\tilde{A}_\alpha$ and represented by a crisp set for all $s \in S$, where $\mu_{\tilde{A}} \geq \alpha$

$$\tilde{A}_\alpha = \{\mu_{\tilde{A}} \geq \alpha, \alpha \in [0,1]\}. \quad (3.26)$$

- **Fuzzy Numbers**

Fuzzy numbers (Dubois & Prade, 1978) are another set in R that is often used to define the concept of a fuzzy set . Fuzzy numbers are described as a subset of real numbers with a membership function $\mu_A(x)$ that is a continuous mapping from R (real line) to a closed interval [0,1]. Triangular and TrFNs are the most used in fuzzy set theory applications. In particular, TFNs are the simplest model of fuzzy numbers and can be easily represented by $\tilde{A}$ = (a_1, a_2, a_3). The parameter a_2 is the maximum possible evaluation data value of

x, The parameters a_1 and a_3 represent the lower and upper bounds of the evaluation data's available area (Lee et al., 2002). The graph of the membership function $\mu_{\tilde{A}}(x)$ in Figure 3.3 has a triangular shape, with the base over the interval $[a_1, a_3]$ and vertex at $x = a_2$.

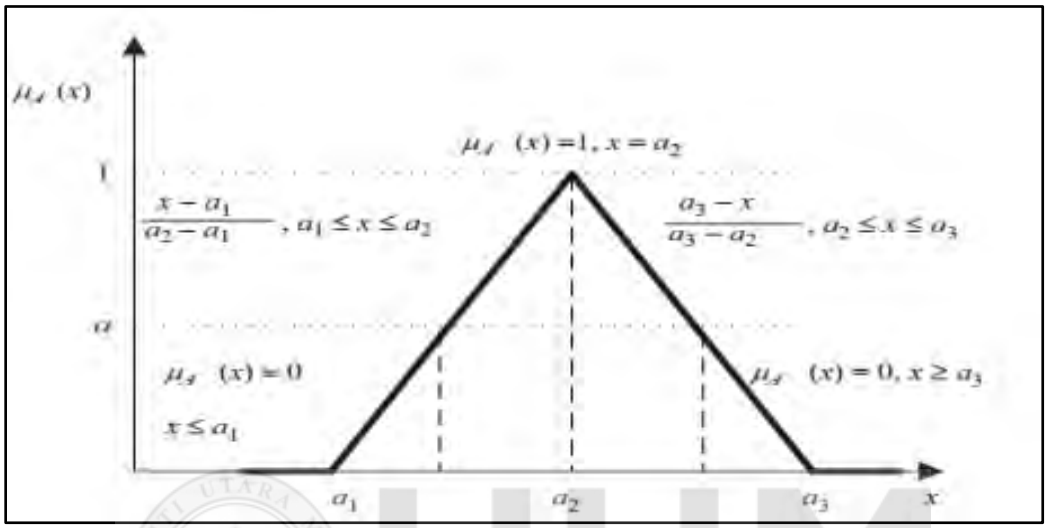


Figure 3.3. Membership function for the triangular fuzzy number $\widetilde{\mathrm{A}} = (\mathrm{a}_1, \mathrm{a}_2, \mathrm{a}_3)$

To be considered a fuzzy number, a fuzzy set $\tilde{A}$ on R must have the following characteristics (Gani & Assarudeen, 2012):

1- $\tilde{A}$ must be a normal fuzzy set, i.e., there is an $x_0 \in R$, in which $\mu_{\tilde{A}}(x_0) = 1$.

2- $\tilde{A}$ is a convex fuzzy set, i.e., $\mu(\lambda x_1 + (1-\lambda)x_2) \geq min\{\mu(x_1), \mu(x_2)\}\ \forall x_1, x_2 \in R, \lambda \in [0,1]$.

3- $\tilde{A}_\alpha$ must be a closed interval for every α ∈ [0,1].

4- The support of $\tilde{A}$ must be bounded, i.e., $\mu_{\tilde{A}}(x)$ is a non-decreasing and a non-increasing bounded left continuous function over [0,1].

Definition 3.5 (Gani & Assarudeen, 2012): TFN It is a fuzzy number described by three points as presented $\tilde{A} = (a_1, a_2, a_3)$. The following conditions are held by this description, which is explained as a membership function.

i. The function from a_1 to a_2 increases.

ii. The function from a_2 to a_3 decreases.

iii. $a_1 \leq a_2 \leq a_3$.

$$\mu_{\tilde{A}}(x) = \begin{cases} 0 & for\ x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1} & for\ a_1 \leq x \leq a_2 \\ \dfrac{a_3 - x}{a_3 - a_2} & for\ a_2 \leq x \leq a_3 \\ 0 & for\ x > a_3 \end{cases}.$$

Definition 3.6 (Banerjee & Roy 2012): The TrFN is an extension of the TFN and is described as $\tilde{A} = (a_1, a_2, a_3, a_4)$, where the membership function:

$$\mu_{\tilde{A}}(x) = \begin{cases} 0 & for\ x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1} & for\ a_1 \leq x \leq a_2 \\ 1 & for\ a_2 \leq x \leq a_3 \\ \dfrac{a_4 - x}{a_4 - a_3} & for\ a_3 \leq x \leq a_4 \\ 0 & for\ x > a_4 \end{cases}.$$

- **Triangular Fuzzy Number Arithmetic Operations**

The four operations that can be performed on TFNs using the function principle are as follows (Banerjee & Roy, 2012): Let $\tilde{A} = (a_1, a_2, a_3)$ and $\tilde{B} = (b_1, b_2, b_3)$, then:

i. Addition: $\tilde{A} + \tilde{B} = (a_1 + b_1, a_2 + b_2, a_3 + b_3)$.

ii. Subtraction: $\tilde{A} - \tilde{B} = (a_1 - b_3, a_2 - b_2, a_3 - b_1)$.

iii. Multiplication:

$\tilde{A} \times \tilde{B} = (\min(a_1 b_1, a_1 b_3, a_3 b_1, a_3 b_3)\,, a_2 b_2, \max(a_1 b_1, a_1 b_3, a_3 b_1, a_3 b_3))$.

iv. Division:

$$\tilde{A}/_{\tilde{B}} = (\min\left({a_1}/{b_1}, {a_1}/{b_3}, {a_3}/{b_1}, {a_3}/{b_3}\right), {a_2}/{b_2}, \max({a_1}/{b_1}, {a_1}/{b_3}, {a_3}/{b_1}, {a_3}/{b_3})).$$

The TFNs are the most straightforward representation of a fuzzy interval, using only the interval $[p_1, p_3]$ of potential values and one single plausible value p_2. The probability for each interval processing time can be easily represented by these three real numbers, shaping the membership function into a triangular shape. Due to its ease of use and simplicity, the TFNs have thus been employed by most researchers to model the processing time (Shao et al., 2020). Processing times and due dates for jobs are inherently uncertain. Hence, they are regarded here alongside TFNs to address the unpredictable nature of parameters in practical contexts. The first fuzzy number showed processing times $\tilde{p}_{ij} = ({p^1}_{ij}, {p^2}_{ij}, {p^3}_{ij})$, where ${p^1}_{ij}$ is the shortest processing time, ${p^2}_{ij}$ is the most possible processing time and ${p^3}_{ij}$ is the longest processing time. To represent job processing times, the following triangular fuzzy membership function is used:

$$\mu_{\tilde{p}_{ij}}(x) = \begin{cases} 0 & for\ x < {p^1}_{ij} \\ \frac{x - {p^1}_{ij}}{{p^2}_{ij} - {p^1}_{ij}} & for\ {p^1}_{ij} \le x \le {p^2}_{ij} \\ \frac{{p^3}_{ij} - x}{{p^3}_{ij} - {p^2}_{ij}} & for\ {p^2}_{ij} \le x \le {p^3}_{ij} \\ 0 & for\ x > {p^3}_{ij} \end{cases}.$$

The second fuzzy number illustrated the due dates $\tilde{d}_j = ({d^1}_j, {d^2}_j, {d^3}_j)$, where ${d^1}_j$ is the earliest due date, ${d^2}_j$ is the most possible due date and ${d^3}_j$ is the latest due date. The following triangular fuzzy membership function is used to represent job due dates:

$$\mu_{\tilde{d}_j}(x) = \begin{cases} 0 & for\ x < {d^1}_j \\ \dfrac{x - {d^1}_j}{{d^2}_j - {d^1}_j} & for\ {d^1}_j \le x \le {d^2}_j \\ \dfrac{{d^3}_j - x}{{d^3}_j - {d^2}_j} & for\ {d^2}_j \le x \le {d^3}_j \\ 0 & for\ x > {d^3}_j \end{cases}.$$

The fuzzy environment impacts the objective functions since the fuzzy completion time of each job will be a TFN due to the triangular form of the processing times. It is defined as a $\tilde{C}_{ij} = ({C^1}_{ij}, {C^2}_{ij}, {C^3}_{ij})$, the tardiness of job j on machine i equals to the $\tilde{T}_{ij} = ({T^1}_{ij}, {T^2}_{ij}, {T^3}_{ij})$, where $\tilde{T}_{ij} = max\ (\tilde{C}_{ij} - \tilde{d}_j, 0)$, where the zero fuzzy number = (0, 0, 0).

The proposed deterministic model has been fuzzified using TFNs for the primary parameters of processing time and due date, namely fuzzy multi-objective UPMSP model, as outlined below:

$$Min\ F = w\widetilde{C_{max}} + (1-w)\sum_{j=1}^{n}\sum_{i=1}^{m}\tilde{T}_{ij}, \tag{3.27}$$

$$\tilde{C}_0 = 0, \tag{3.28}$$

$$\tilde{C}_{ij} - \tilde{C}_{ik} + v(1 - X_{ijk}) \ge \tilde{p}_{ik} \quad \forall i \in M, \forall k \in N_0, \forall j \in N: j \ne k, \tag{3.29}$$

$$\widetilde{C_{max}} \ge \tilde{C}_j \quad \forall j \in N, \tag{3.30}$$

$$\tilde{T}_j \ge \tilde{C}_j - \tilde{d}_j \quad \forall i \in M, \forall \mathrm{j} \in N, \tag{3.31}$$

$$\tilde{p}_{ij}, \tilde{C}_{ij}, \tilde{T}_{ij}, \widetilde{C_{max}} \ge 0. \tag{3.32}$$

3.2.2.2 Defuzzification Method

In a scheduling algorithm, the ranking of two numbers is critical. When all the inputs are real numbers, ranking them is simple. However, in such a fuzzy environment, it is challenging to determine whether a job will be completed on time or late (Shen, 2019). To determine the maximum fuzzy makespan and total tardiness, it is necessary to adopt a method to compare TFNs (Li et al., 2020). In addition, the process involves using the total importance expected values of fuzzy numbers to deal with fuzziness (Peddi, 2019). It is also called the defuzzification process, which converts a fuzzy output to a single or crisp output value (Chakraverty et al., 2019).

Liao and Su (2017) presented an in-depth review and comparison of the performance of nine fuzzy ranking methods to study the effect of fuzziness on the fuzzy parameters of processing time, release time, and setup time to minimize makespan for UPMSP. Based on extensive testing, they noted that the total integral value method for ranking fuzzy numbers developed by Liou and Wang (1992) outperformed the other ranking methods.

To convert the proposed fuzzy mathematical model of UPMSP to a crisp model, the total integral value method is used to compare TFN $A = (a, b, c)$, which can be represented as:

$$I_T^{\lambda}(A) = \frac{1}{2}\lambda(b + c) + \frac{1}{2}(1 - \lambda)(a + b),$$

$$= \frac{1}{2}[\lambda c + b + (1 - \lambda)a],$$

where λ represents a decision maker's degree of optimism and $\lambda \in [0, 1]$.

Thus, besides representing the two most basic and frequently conflicting objectives in production scheduling, the decision to use makespan and total tardiness was deliberate. They simply adopted the multi-objective that Kongsri and Buddhakulsomsiri's model

proposed as they are. Next, the fuzzy aspect was dealt with directly using TFNs for the processing time, due date, and consequently, the objective function. To deal with the proposed model's fuzzy nature, a defuzzification method must be used to compare TFNs and defuzzify the resulting fuzzy multi-objective makespan and total tardiness to simplify the fuzzy output to a single crisp number.

Building on the phase 2, the revised mathematical UPMSP model successfully demonstrates that the selected objective, makespan and total tardiness, accurately reflect the fuzzy multi-objective scheduling theoretical construct. To solve the proposed problem, the next phase will concentrate on modifying AFSA and adapting it to solve the discrete UPMSP model.

3.2.3 Phase 3: Modify AFSA Algorithm

The main objective of this phase is to have a modified AFSA algorithm to increase the effectiveness of standard AFSA behaviors and make it appropriate for solving the proposed problem. The standard AFSA has been modified to solve fuzzy multi-objective UPMSP to minimize makespan and total tardiness in three aspects. The first aspect is increasing the effectiveness of standard AFSA by proposing new behavior. The second aspect ensures an equilibrium among global search capability and convergence rate by specifying the specific requirements for the main algorithm parameters. The third aspect makes the continuous AFSA algorithm appropriate for solving discrete UPMSP and improves the original algorithm's search capabilities using the transformation method. Validating the translation of continuous AFSA to discrete AFSA was done by testing various encoding schemes in order to successfully translate the continuous movement of artificial fish into meaningful discrete actions for job scheduling. An ineffective

transformation method can result in poor solution quality or slow convergence. A detailed explanation of the modification process is discussed in Section 4.2.

3.2.4 Phase 4: Hybridization of the Proposed Modified Algorithm

A hybrid algorithm that combines the proposed modified AFSA with the neighborhood search algorithm was developed in order to improve it and provide more accurate results. AFSA is naturally good at exploration; however, its standard operators are not always effective at adapting a solution to find the absolute best point within a search space. Meanwhile, the neighborhood search algorithm is designed for exploitation. It begins with a single solution and iteratively explores its immediate neighbors to discover a better one. By employing both algorithms, AFSA functions as a global search method, identifying promising regions within the vast search space. The neighborhood search algorithm serves as a local search technique, refining the promising solutions discovered by AFSA to obtain the best solution within the local region. The hybridization of the proposed modified AFSA is based on the three neighborhood operations employed: swap, reversion, and insertion between the locations of two random jobs in the same solution to increase intensity in a promising region, thus improving algorithm performance. Therefore, choosing swap, reversion, and insertion over many other operations creates a powerful combination because these operations provide simple calculations and produce significant changes that greatly improve the final solution (Li and Yin, 2013). Section 4.3 discusses the hybridization process in detail.

3.2.5 Phase 5: Evaluation of the Proposed Algorithms

This phase involves two sections of comparison and analysis that illustrate the performance evaluation of the proposed algorithms. Notably, three different-sized problem instances are randomly generated to simulate the experimental design of the proposed problem. Using MATLAB simulation, randomly generated computational experiment sets of job and machine combinations of small, medium, and large sizes are used to analyze the problem size effect, with ten instances for each size. The algorithms are validated using randomly generated data that attempts to simulate real-world scenarios. The range selection method is inspired by (Lin & Lin, 2015). A small-sized problem with N = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50] and M = [3, 3, 4, 4, 5, 5, 6, 6, 7, 7]; a medium-sized problem with N = [60, 80, 100, 120, 140, 160, 180, 200, 220, 240] and M = [8, 8, 9, 9, 10, 10, 11, 11, 12, 12]; and a large-sized problem with N = [250, 270, 290, 310, 330, 350, 370, 390, 410, 430] and M = [13, 13, 14, 14, 15, 15, 16, 16, 17, 17].

The results of the proposed modified and hybrid algorithm will be compared based on Minimum (Min), Maximum (Max), mean, and Standard Deviation (STD), as well as Wilcoxon signed-rank test statistic values.

The four performance evaluation criteria, i.e. minimum, maximum, mean, and standard deviation are insufficient to assess the proposed algorithms or it gives close values in some algorithms. Therefore, the hypothesis testing should be performed by using statistical tests like the Wilcoxon signed-rank test for non-parametric data to ascertain whether observed performance differences are statistically significant or merely the result of chance.

The first phase involves verifying the efficacy of the proposed modified AFSA algorithm by randomly generating small, medium, and large-sized instances for testing. The computational experiment results are compared based on Min, Max, mean, and STD, as

well as Wilcoxon signed-rank test statistic values, with the results obtained by the standard AFSA algorithm and five different variations of modified AFSA algorithms.

The second phase validates the effectiveness of the proposed hybrid AFSA algorithm by assessing and comparing the computational experiment results with three sets of data. The first set used the AFSA algorithm and five modified versions of AFSA. Consequently, with the proposed modified AFSA algorithm and five different metaheuristics algorithms, Simulated Annealing (SA), Tabu Search (TS), Firefly Algorithm (FA), Genetic Algorithm (GA), Particle Swarm Optimization (PSO). Finally, three existing hybrid algorithms were used on the same three-sized problem instances and the same performance criteria.

3.2.5.1 Performance Evaluation Criteria

The performance evaluation criteria are significant elements in determining the proposed algorithm's performance. Performance evaluation criteria used for the entire computational tests include Min, Max, mean, and STD. Consequently, the Wilcoxon signed-rank statistical test was also conducted, and the comparison of the test values was considered as follows:

a. **Minimum (Min)**: represents the lowest value obtained by the defined objective function in the process of repetition. In minimization problems, the minimum value is used to determine the best performance of the algorithm.
b. **Maximum (Max)**: represents the highest value obtained by the defined objective function in the process of repetition.
c. **Mean**: the statistical measure produced by the function during the repetition process, which is calculated by dividing the total number of observed outcomes from the set of results by their sum and defined as follows:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{3.33}$$

d. Standard deviation (STD): is a crucial statistical metric used to quantify the variance or disarray of a data set for function values and is defined as follows:

$$\text{STD} = \sqrt{\frac{1}{n-1}\sum_{I=1}^{n}(x_i - \bar{x})^2}. \tag{3.34}$$

e. Wilcoxon Signed-Rank Test: is a nonparametric statistical test that has become a popular computational intelligence approach for improving the process of evaluating a new method's performance within the framework of experimental analysis to determine which algorithm is superior to another (Derrac et al., 2011). This test's advantage is the ability to assess two related and matched samples without being aware of the distribution of the data (Li et al., 2021). This test provides answers to the following hypotheses:

$$H_0: mean(A) = mean(B),$$

$$H_1: mean(A) \neq mean(B),$$

where A and B represent the results of the first and second algorithms, respectively. The null hypothesis for this test notes no significant difference in performance among paired algorithms. Meanwhile, the alternative hypothesis states that "there is a significant difference among paired algorithms in performance." Assume Indicate (Ind) to demonstrate the difference in two algorithms' performance scores when solving i^{th} out of n problems. If Ind = +1, it indicates that the first algorithm is superior to the second. Conversely, if Ind = −1, it suggests that the second algorithm is superior to the first algorithm. Let R+ represents the total ranking indicating that the first method is superior to the second one, while R− represents the total ranking indicating that the second method

is superior to the first one, the property $R^+ + R^- = \frac{n\times(n+1)}{2}$ must be true. Ranks of $\text{Ind} = 0$ are distributed equally between the sums. One is disregarded if there is an odd number of them:

$$R^+ = \sum_{\text{Ind}>0} rank(\text{Ind}) + \frac{1}{2}\sum_{\text{Ind}=0} rank(\text{Ind}),$$

$$R^- = \sum_{\text{Ind}<0} rank(\text{Ind}) + \frac{1}{2}\sum_{\text{Ind}=0} rank(\text{Ind}),$$

p-value are the assessment indexes between zero and one. The *p*-value indicates the significance level of the hypothesis test, and it can also indicate how significant the result is, i.e., the smaller the *p*-value, the greater indication against the null hypothesis. We used MATLAB to calculate *p*-value to compare algorithms at a significance level of $\alpha = 0.05$, and when ($R^+ = \frac{n\times(n+1)}{2}$), then this algorithm is superior to all algorithms through every test.

3.2.5.2 The Comparative Modified Algorithms

The advantages of the proposed modified algorithm and the effectiveness of the modification process are demonstrated using five different versions of the modified AFSA as comparison algorithms alongside the standard AFSA proposed by Li (2002). These include Modified 1 (Zhao et al., 2023), Modified 2 (Tan & Mohamad-Saleh, 2020), and Modified 3-Method 1, Modified 3-Method 2, and Modified 3-Method 3 proposed by Huang et al. (2021). In these algorithms, the modification process is included in the visual and step parameters as follows:

1. The first modified algorithm (Modified 1) introduced an improved parameters for visual and step. The visual parameter is calculated by the weighted average method using the following formula:

$$Visual_i = \frac{\sum_{i=1,i\neq j}^{N} D_{ij} \times w_{ij}}{\sum_{i=1,i\neq j}^{N} w_{ij}}, \quad (3.35)$$

where N is the total number of AFs, each AF is recorded as F_i (i = 1, 2, 3, ..., N), and D_{ij} denotes the distance between two AFs F_i and F_j. The weight factor of the distance D_{ij} is w_{ij}.

Furthermore, the improved parameter for the step parameter is calculated as follows:

$$Step = Step \times (1 - \frac{i}{MaxIter}), \quad (3.36)$$

where i denotes the current iteration, and MaxIter is the total iteration.

2. The second modified algorithm (Modified 2) introduced an improved parameters, visual, and step, using equations 4.2,4.3, and 4.4 as discuss in Section 4.2.
3. The third modified algorithm (Modified 3) focused on the influence of the step parameter of the algorithm. In particular, it adapted three improved step methods to use in preying, swarming, and following behaviors in the algorithm. These methods are presented as follows:

Method 1: $Step_{t+1} = \left(1 - \frac{t}{\alpha_1 \times I}\right) \times Step_t,\ Step_1 = Step,$ (3.37)

Method 2: $Step_{t+1} = \alpha_2 \times step_t, Step_1 = Step,$ (3.38)

Method 3: $Step_{t+1} = step_t + \frac{step_I}{\alpha_3 \times I},\ Step_1 = Step,$ (3.39)

where I is the total number of iterations, t is the number of iterations, *Step* is a fixed parameter, and α_1, α_2, α_3 are the attenuation factors of the three improved steps. Correspondingly, the three methods modified three behaviors as follows:

- Modified 3-Method 1: apply three improved step methods on the equation of preying behavior, and described as:

$$X_i(t+1) = X_i(t) + \frac{X_j - X_i}{\| X_j - X_i \|} \times \left(1 - \frac{t}{\alpha_1 \times I}\right) \times step_t \times R(0,1), \quad (3.40)$$

$$X_i(t+1) = X_i(t) + \frac{X_j - X_i}{\| X_j - X_i \|} \times \alpha_2 \times step_t \times R(0,1), \quad (3.41)$$

$$X_i(t+1) = X_i(t) + \frac{X_j - X_i}{\| X_j - X_i \|} \times \left(step_t + \frac{step_I}{\alpha_3 \times I}\right) \times R(0,1). \quad (3.42)$$

- Modified 3-Method 2: apply three improved step methods on the equation of swarming behavior, and described as:

$$X_i(t+1) = X_i(t) + \frac{X_{center} - X_i}{\|X_{center} - X_i\|} \times \left(1 - \frac{t}{\alpha_1 \times I}\right) \times step_t \times R(0,1), \quad (3.43)$$

$$X_i(t+1) = X_i(t) + \frac{X_{center} - X_i}{\|X_{center} - X_i\|} \times \alpha_2 \times step_t \times R(0,1), \quad (3.44)$$

$$X_i(t+1) = X_i(t) + \frac{X_{center} - X_i}{\|X_{center} - X_i\|} \times \left(step_t + \frac{step_I}{\alpha_3 \times I}\right) \times R(0,1). \quad (3.45)$$

- Modified 3-Method 3: apply three improved step methods on the equation of the following behavior, and described as:

$$X_i(t+1) = X_i(t) + \frac{X_j - X_i}{\|X_j - X_i\|} \times \left(1 - \frac{t}{\alpha_1 \times I}\right) \times step_t \times R(0,1), \quad (3.46)$$

$$X_i(t+1) = X_i(t) + \frac{X_j - X_i}{\|X_j - X_i\|} \times \alpha_2 \times step_t \times R(0,1), \quad (3.47)$$

$$X_i(t+1) = X_i(t) + \frac{X_j - X_i}{\|X_j - X_i\|} \times \left(step_t + \frac{step_I}{\alpha_3 \times I}\right) \times R(0,1). \quad (3.48)$$

For $t = 1, 2\ldots I\text{-}1$, when the number of iterations is $t + 1$.

The parameters of these algorithms are set to be equivalent to those of the proposed algorithm for a fair comparison. Each algorithm is executed ten times to assess the variation in the generated solutions. A thorough comparison between these five versions of the modified AFSA and the proposed modified algorithm is provided in Section 5.3.

3.2.5.3 Comparative Single Standard Metaheuristic Algorithms

The five metaheuristic algorithms involved in the comparison are the Firefly Algorithm (FA), Tabu Search (TS), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA) algorithms. The brief introduction of these algorithms is discussed in the following:

i. Firefly algorithm

Inspired by the patterns and behaviors of fireflies, Yang (2008) created a new metaheuristic algorithm called the FA. It is a highly efficient population-based algorithm in the swarm intelligence domain. It outperforms single-search algorithms and can be used to solve most optimization problems. According to Kumar and Kumar (2021) comprehensive review of FA, the FA performs better than many algorithms and can solve Nondeterministic Polynomial Time (NP-hard) optimization problems. In particular, FA has two notable variables: light intensity and attractiveness. The main feature of fireflies

is their flashlight. It can be used to attract potential partners and warn against possible dangers or enemies. According to the law of physics, the intensity of a flashlight I decreases as the distance r increases (Kumar & Kumar, 2021):

$$I \propto_1 / r^2. \tag{3.49}$$

Consequently, the firefly blinks its flashlight at regular times using $\theta = 2\pi$. When a firefly moves into the neighborhood, it forms a couple. This behavior is inspired by a graph coloring problem (Fister et al., 2013). Attractiveness is directly proportional to the light intensity perceived by other fireflies. The pseudocode of the FA Algorithm is presented in Figure 3.4

Result: Best Solution Available
Let there are *npop* initial population solutions and i denotes the number of iteration, with max-it maximum iteration number. I is the intensity or brightness of flash;
Generate initial solution *npop* randomly.;
for $i \leftarrow$ 1 to max-it do
  Calculate Brightness I;
  sort I such as $I_j \geq I_{j-1}$, for each j;
  for $j \leftarrow 1$ to $N - 1$ do
    for $k \leftarrow j + 1$ to N do
      if $I_k \geq I_j$ then
        Firefly j move in the direction of Firefly k;
      end
    end
  end
  Move firefly in random manner;
end
Store the best solution

Figure 3.4. The pseudocode of the Firefly Algorithm

ii. Tabu search algorithm

Based on local search techniques, Tabu Search (TS) is a popular metaheuristic search solution for optimization problems. It starts with an initial solution and iteratively explores potential solutions to identify the optimal solution (Glover, 1986). Notably, most applications for TS have been and still are in discrete optimization (Laguna, 2018). A comprehensive survey of TS by Prajapati et al. (2020) over the last two decades reveals that TS algorithms have been used for various real combinatorial optimization problems. This includes applications linked to certainty or uncertainty scheduling problems. Since the TS neighborhoods are formed by the rules connected with problems, which results in a smaller solution space, the TS algorithm has the highest efficiency among the metaheuristic algorithms (Chen et al., 2022). The pseudocode of the TS algorithm is illustrated in Figure 3.5.


```
Begin

T: =[];
S:= initial solution;
S* := S
Repeat
    find the best admissible S' ∈ N(s);
    if f(S') > f(S*) then S* := S'
     S := S';
      update tabu list T;
       until stopping criterion;
  end
```

Figure 3.5. The pseudocode of TS Algorithm

iii. Genetic algorithm

The GA is a probability search method that was created by John Holland in 1970, inspired by the principles of natural selection and genetics. A chromosome represents a solution in GA, which begins with a population of solutions. Throughout each generation, the population size remains constant. At each generation, the fitness of each chromosome is evaluated, and chromosomes for the next generation are probabilistically chosen based on their fitness values. The evolutionary process is repeated until the final condition is met. Furthermore, GA employs methodologies derived from evolutionary biology, including inheritance, mutation, selection, and crossover. This aims to identify optimal solutions for complex problems across diverse fields such as biology, engineering, computer science, and social science (Kumar et al., 2010). The GA pseudocode is displayed in Figure 3.6.


```
Begin
Initial population
While (generation ≠ termination)
    Evaluate population
    Selection
    Crossover
    Mutation
  Replacement
    Output best chromosome
end
```

Figure 3.6. The pseudocode of GA Algorithm

iv. Particle swarm optimization

PSO was first introduced by Kennedy and Eberhart in 1995. The social behaviors of bird flocks inspired the concept and development of the PSO algorithm to solve optimization problems, where the flock is viewed as a swarm of particles, with each particle representing a candidate solution. Every particle flies through a predetermined search space to identify its optimal solution, adjusting its movement based on its own and other particles' flying experiences. The PSO algorithm is an appealing optimization method due to its straightforward implementation. It only requires three control parameters (inertia weight, cognitive ratio, and social ratio) and is sufficiently flexible to be hybridized with other optimization algorithms. In other words, PSO is an effective tool for solving various optimization issues in numerous real-world applications. Based on a comprehensive review, some instances include scheduling, feature selection, wireless communications, image processing, and electrical power systems (Shami et al., 2022). The pseudocode of the PSO algorithm is provided in Figure 3.7.

```
Begin
   Determine the size of the swarm
for j = 1 to size of the swarm do
   Generate an initial position and velocity of the particle,
   Evaluate the particle;
   Determine the pbest (personal best position) of the particle
end
Select the gbest (the best position) found in the swarm
  while termination criterion is not met do
 for j = 1 to size of the swarm do
      Update the velocity and position of each particle
      Evaluate new position of the particle;
      Update the pbest (personal best position) of each particle
end
Update the gbest (the best position) found in the swarm
end
```

Figure 3.7. The pseudocode of the PSO Algorithm

v. Simulated annealing algorithm

SA is a popular global optimization algorithm first developed by Metropolis et al. (1953). It simulates the annealing process of solids and draws its motivation from statistical physics, in which an artificial temperature controls the exploration of the search space while preserving convergence to global minima. This procedure creates high-quality materials by gradually cooling them at a high temperature. As a result, the initial solution is generated randomly using a high temperature. The temperature gradually decreases until the optimal solution is achieved (Delahaye et al., 2018). Figure 3.8 depicts the pseudocode for the SA algorithm (Pirim et al., 2008).

Begin

S:=Initial solution
k:=1;
repeat
  generate an $s' \in N(s)$;
  if $f(s') \geq f(s)$ **then** s:= s'
  else
    if $exp\left(\frac{f(s')\geq f(s)}{c_k}\right) > random[0,1)$ **then** s:= s';
  k:=k+1;
until stop criterion:
end;

Figure 3.8. The pseudocode of SA Algorithm

3.3 Chapter Summary

Briefly, this chapter discusses the methodology of the work by illustrating the research process, which comprises five phases. The definition of the problem includes the background information and a review of the relevant work. A mathematical model is

developed by revising upon the model used by Kongsri and Buddhakulsomsiri (2020) to enhance its generality. This continues by presenting the main definitions of fuzzy set theory and arithmetic operations for TFNs and the critical foundation for all concepts required to aid comprehension. A fuzzy mathematical model then constructed using fuzzy triangular membership functions for the main parameters, i.e. processing time, and due date. The standard AFSA algorithm is modified in three aspects to make it suitable for the proposed problem. Subsequently, the hybrid algorithm of AFSA is developed using the proposed modified AFSA algorithm and neighborhood search algorithm to improve the current modified AFSA and obtain more precise solutions. Finally, the evaluation phase involves the validation of the effectiveness of the proposed algorithms.

CHAPTER FOUR

MODIFIED ARTIFICIAL FISH SWARM ALGORITHM AND ITS HYBRIDISATION

4.1 Introduction

The main contribution to achieving our research objective of developing algorithms to solve the fuzzy multi-objective Unrelated Parallel Machine Scheduling Problem (UPMSP) is discussed in this chapter. The discussion is divided into two phases. The first phase focuses on developing the proposed modified artificial fish swarm algorithm (AFSA) (Section 4.2). The second phase includes creating a hybrid algorithm that combines the proposed modified algorithm and a neighborhood search algorithm (Section 4.3). The summary of this chapter is presented in Section 4.4.

4.2 The Proposed Modified AFSA Algorithm

In this section, the proposed algorithm was modified in three aspects: aspiration behavior, adaptive step and visual parameters, and the use of a transformation method. These modifications aim to increase the effectiveness of the standard AFSA behaviors and adapt the algorithm to better fit the proposed problem. The first aspect is an aspiration behavior that is inserted into the four major AFSA behaviors to avoid random searches of AF in its visual range for the next state prior to moving to the next position. This suggests that AF should try all possible directions until the position is determined, and this position generated may not be optimal since the process is random. Consequently, this causes drawbacks in the AFSA's algorithm behaviors and consumes much time.

After all AFSA behaviors have been executed, the best position of AFs is listed on the bulletin board, and this serves as inspiration for aspiration behavior. Therefore, X_{best} is supposed to indicate the best position of AF in the visual range and $step$, such that

$\| X_{best} - X_i \| \leq step$. Thus, it can be stated as:

$$X_{Aspiration} = X_i + \frac{X_{best} - X_i}{\| X_{best} - X_i \|} \times step2 \times R(0,1). \quad (4.1)$$

The aspiration behavior aims to speed up the convergence of the algorithm by allowing an individual AF to learn from the best-performing fish in the swarm. Ensures the AF can determine the best position in one calculation cycle, significantly reducing computation time. Figure 4.1 shows the pseudocode of the proposed aspiration behavior.

```
Adapt the best solution found ever on the bulletin board
Calculate fitness values at the new position

  Within improved visual parameter
    Move along a particular direction based on equation (4.1) within improved step parameter
    to select the best optimal solution and update the current position of AF

        If the solution value gets no improvement
            Stay at the best position, and best solution is the final optimal solution
        Else

        Move to the new position and consider it a global optimal solution

        Update the current best solution of AF in the bulletin board.
```

Figure 4.1. Pseudocode of the aspiration behavior

The following is a more detailed description of the steps involved in the aspiration behavior as shows in figure 4.1:

"Adapt the best solution found ever on the bulletin board",

"calculate fitness value at the new position"

1. Find global best: The current fish (AF_i) determines the global best position (X_{best}) by examining the bulletin board and calculate the fitness value of current and global best position.

"Within the improved visual and step parameters"

2. Calculating the aspiration move: which changes dynamically based on equations 4.2 and 4.3 to adjust the search. The fish takes a step from its current position (X_i) towards (X_{best}) to calculate a new potential position ($X_{Aspiration}$) based on equation 4.1.

" Move along a particular direction based on equation (4.1) within the improved step parameter to select the best optimal solution and update the current position of AF"

3. Movement evaluation: To select the best solution and update the current position of AF, the algorithm compares the fitness of $X_{Aspiration}$ with the fitness of (X_{best}). The move was ineffective if the fitness of $X_{Aspiration}$ is not better than the fitness of (X_{best}).

" If the solution value shows no improvement,

Stay in the best position, and the best solution is the final best solution"

4. Discards $X_{Aspiration}$ and stay in the best position as the final best solution if the move failed.

" Else

Move to a new position and consider it a global best solution

Update the current best solution for AF on the bulletin board"

5. Move to a new position and consider it a global best solution if the move was a success, where the fitness of $X_{Aspiration}$ is better than the fitness of (X_{best}). The fish updates the position, X_{best} is now set to $X_{Aspiration}$.

Visual and step are two fundamental AFSA parameters that directly regulate the basic equilibrium between exploration and exploitation. A researcher can develop a better AFSA algorithm that explores broadly by making these two parameters adaptive (starting large and then decreasing over time). Other parameters, such as the crowd factor, the number of fish, and the try number, are less frequently changed because they are more concerned with the swarm's static structure rather than its dynamic search behavior and have a less direct impact on the critical exploration and exploitation balance.

Through iterative processes, the visual and step parameters in the standard AFSA are fixed. The algorithm's convergence speed is increased at the beginning with large visual and step values. However, when AFs reach the final position, the large values will cause issues like local optimum or iterative jumps. Note that effectiveness reduces if the values are excessively low (Zhao et al., 2023; Tan & Mohamad-Saleh, 2020). Hence, the second aspect focuses on improving both the global search capability and the convergence rate by applying specific adjustments to the step and visual parameters. The algorithm's global search capability and convergence speed are improved in the early stages using relatively large visuals and steps. Along the iterations, these parameters are decreased using the following formulas:

$$visual^{t+1} = visual^{t} - visual^{t} \times \lambda + visual_{min}, \tag{4.2}$$

$$step^{t+1} = step^{t} - step^{t} \times \lambda + step_{min}, \tag{4.3}$$

$$\lambda = e^{\left(-\frac{\sigma}{\sqrt[4]{(t_{max})^3}} * (t_{max} - t)\right)}, \tag{4.4}$$

where $visual_{min}$ is the minimum visual value, $step_{min}$ is the minimum step value, $t = (1, 2, \ldots, t_{max})$ is the index number of iterations, and $0.5 < \sigma < 1$ at any stage. Therefore, it improves the algorithm's local search at a later stage.

It is apparent from the above equations that the visual and step are gradually decreasing throughout the iterations prior to dropping to a value that roughly corresponds to $visual_{min}$and $step_{min}$, respectively. Indeed, $visual_{min}$ and $step_{min}$ act to limit the iterative variation of the step and the visual, respectively. Accordingly, visual and step will eventually decrease to almost zero without the presence of $visual_{min}$ and $step_{min}$. When this occurs, AFs are unable to make any significant movement and thus cannot conduct any additional local searches.

In optimization algorithms, the solution encoding is a crucial process. Since AFSA was originally a continuous algorithm designed to solve continuous problems, it is challenging to solve combinatorial optimization problems such as UPMSP with a discrete solution space. Therefore, making the AFSA algorithm suitable for solving discrete UPMSPs and improving the original algorithm's search capabilities is crucial. The third aspect in the proposed algorithm is to use transformation method to transform the continuous solution into a discrete solution. For a problem with N jobs and M machines, the sequence of jobs on each machine is determined by encoding the real schedule solution (continuous solution) of the AFSA algorithm. This is by indexing jobs and sorting the fractional parts in ascending order to transform a real number series into an integer series. Therefore, the

integer series is a discrete solution. Figure 4.2 shows the pseudocode for the transformation method.

```
function [a]=trans(q)
   n=length(q);
   s=random('uniform',0.1,0.2,1,1);
for i=1:n
  a(q(i))=s;
  s=s+random('uniform',0.1,0.2,1,1);
end
a=a./norm(a);
```

Figure 4.2. Pseudocode of the transformation method

This code uses a trans function to transform the input vector *q,* representing the continuous solution, into the output vector *a,* representing the discrete solution. It starts by:

Step 1: Determine the length of *q* to identify the number of elements in *q* and store it in *n*.

Step 2: Initialize *s* with a random value generated from a uniform distribution in the range [0.1, 0.2]. The assignment process will begin with this value.

Step 3: Loop through each element of *q*. For every element in *q,* perform the following steps inside the loop:

- $a(q(i)) = s$: The current value of *s* would be assigned to the position in *a* for which $q(i)$ is specified. This step maps the continuous solution to a discrete position.
- $s = s$ + random ('uniform', 0.1, 0.2, 1, 1): A new random number from the same uniform distribution is added to update *s*. Thus, the solution's randomness is maintained, and every next assignment in *a* is based on a new random value.

Step 4: Normalize the vector *a* to ensure its Euclidean norm (length) equals one. This step checks that the final discrete solution is appropriately scaled.

Let us explain the proposed method with the following example. Let a UPMSP with five jobs and three machines, and let the lower bound for the algorithm search space be equal to 0 and the upper bound equal to 1. Let the produced solution by AFSA, S = [0.23 0.43 0.1 0.7 0.2], be a continuous solution and index the solution by integer series N = [1 2 3 4 5]. After a combination of solution and index, the solution turns out to be:

1	2	3	4	5
0.23	0.43	0.1	0.7	0.2

continuous solution

Now, we sort the solution by increasing the order in the first row and arranging the index according to the solution. After sorting, the corresponding solution should be:

3	5	1	2	4
0.1	0.2	0.23	0.43	0.7

discrete solution

Thus, the discrete solution D = [3 5 1 2 4].

The proposed modified AFSA is chosen and adapted to deal with the three core complexities of solving the discrete fuzzy multi-objective UPMSP. The AFSA movement equations are inherently continuous. The construction adapts by transforming the continuous movement of artificial fish into meaningful discrete actions for job scheduling. This ensures that each step in the AFSA search space produces a valid job sequence (a discrete solution). Triangular fuzzy numbers (TFNs) are used to apply fuzzy processing times and due dates in the proposed scheduling model. Then transforms the resulting fuzzy makespan and tardiness into crisp values using the total integral value method, a

defuzzification method. These crisp values are then used by the AFSA for comparison and movement decisions. The adaptive visual and step parameters boost suitability by balancing search intensity, allowing for broad exploration early in the search and focused exploitation closer to convergence. As a result, it is a highly suitable search algorithm, whereas exact methods fail to adequately deal with these complexities.

- **The Flow Chart and Pseudocode of Proposed Modified AFSA**

The proposed modified AFSA algorithm is illustrated in Figures 4.3 and 4.4 as a flowchart and pseudocode, respectively. The population size of AFs was set up along with the parameters for the AFSA and the proposed algorithm. After each iteration, each AF is evaluated individually, and the best fitness is recorded on the bulletin board by employing the three behaviors. In particular, the yellow color emphasized in the flowchart indicates the modification steps implemented in the proposed modified AFSA. It started by executing aspiration behavior by identifying the best solution after the execution of these behaviors and enhancing the search process through the comparison of the aspiration state with the current state of the fish. Furthermore, the proposed algorithm employs preying behavior when the conditions for executing behaviors are unmet or when no improvement is seen. Subsequently, the AFSA's fixed value will be substituted with improved visual and step parameters, and the performance will be revised according to the best and contemporary individual employing aspiration behavior expression. The fitness value of the best individual is then determined through a transformation method that converts the continuous solution into a discrete solution. Correspondingly, all solutions will be updated according to the previous steps and performed until the termination criterion is satisfied. The three proposed modification steps, which are interconnected and depend on the

solution's quality, will be executed concurrently to optimize the overall process, concentrating on the main weaknesses of the standard AFSA.

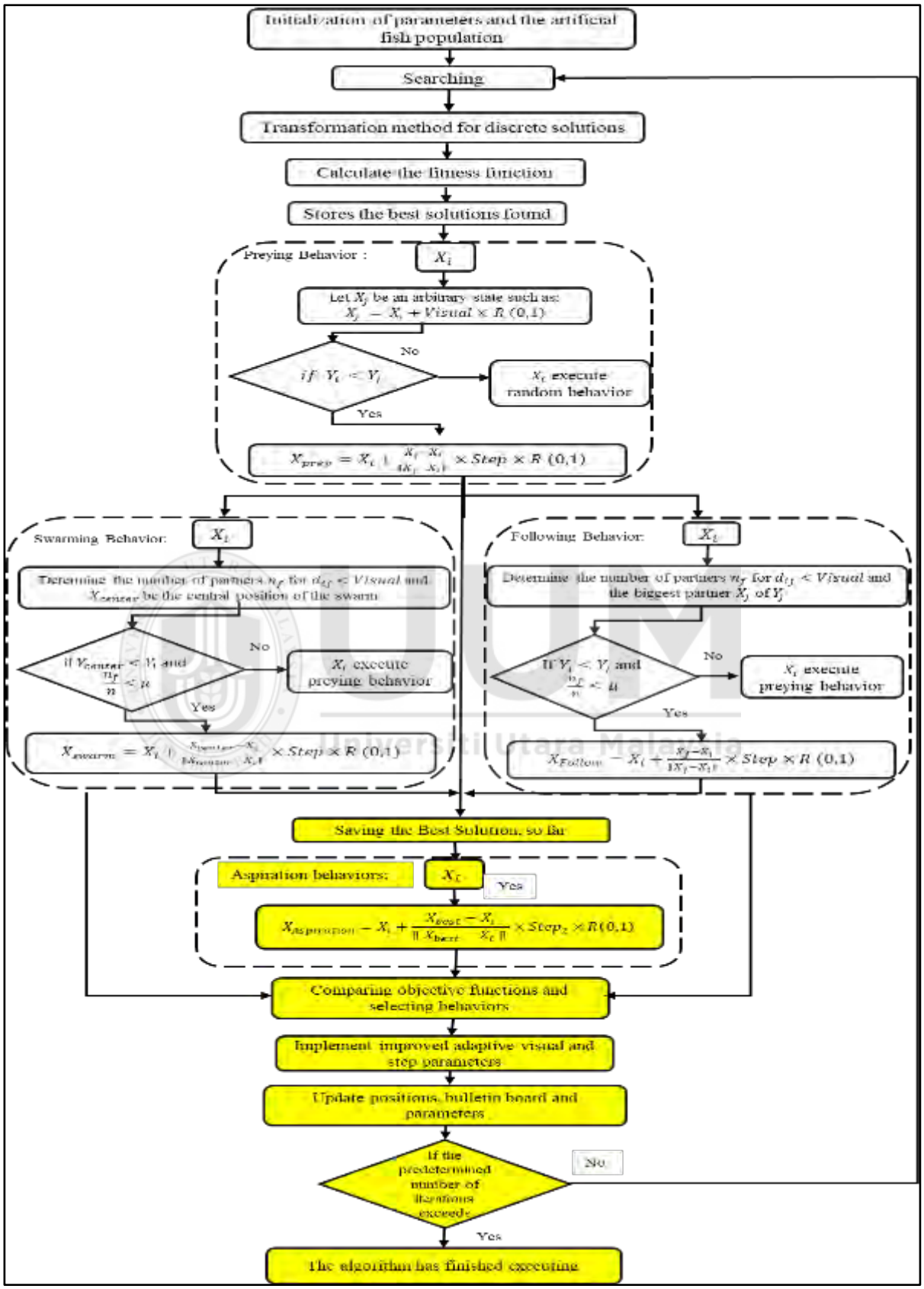


Figure 4.3. Flowchart of the proposed modified AFSA algorithm

input: npop Number of Fish in the swarm
MaxIt Maximum number of iterations for optimization
Visual The visual range radius
Step The maximum step length that a fish can take at each movement
try-number The number of attempts in preying behavior, μ The crowding factor
output: Optimal fish position and its fitness
Initialize a population of n fish' positions at random
;
While *the stopping condition not satisfied,*
foreach $\mathbf{AF}_i$ *do*
Compute the visual range
if *Visual range is empty* ***then***
$f(X_i) \longleftarrow \text{random}(X_i)$
else
if *Visual range is crowd* ***then***
$f(X_i) \longleftarrow$ *follow* (X_i)
else
Calculate Swarm centroid X_{center}, **if** Fitness of X_{center} $(f(X_{Swarm}))$ *better than Fitness of* AF_i ***then***
$f(X_i^1) \longleftarrow$ swarm (X_i)
else
$f(X_i^1) \longleftarrow$ follow (X_i)
end
if *Fitness of* AF_i *worse than global best* ***then***
$f(X_i^2) \longleftarrow$ pray (X_i)
else
$f(X_i^2) \longleftarrow$ follow (X_i)
end
$f(X_i) = argmin\ fitness\ (f(X_i^1), f(X_i^2))$
end
end

foreach AF_i **do**
$X_i \longleftarrow$ select $(X_i, f(X_i))$
if X_{best} is the best position found ever in the *global best for t Iterations* ***then***
$f(X_{best}) \longleftarrow$ *aspiration* (X_{best})
else
Randomly choose a fish X_i and $f(X_i) \longleftarrow \text{random}(X_i)$
end

end

Figure 4.4. Pseudocode of the proposed modified AFSA algorithm.

The proposed modified algorithm provides three main theoretical implications that improve the effectiveness of the standard AFSA and enhance its suitability for the proposed problem. The first theoretical implication is that the aspiration behavior tries to prevent AF from randomly searching in its visual range for the next state before proceeding to the next position. This proposes that AF should investigate every possible direction until the position is determined. This position may not be optimal since the process is random. It led to drawbacks in the AFSA's behavior and cost much time. Aspiration behavior is stimulated by the best position of AFs, which has been determined and recorded on the bulletin board after executing all AFSA behaviors.

The second theoretical implication is that, in the AFSA, the AF looks for a local solution within its visual range, whereas the fish swarm collaborates to achieve the global optimal solution. As a result, visuals and steps control the balance of local and global search (Pourpanah et al., 2023). Through particular equations for visual and step parameters, the algorithm optimizes its local search in later stages to achieve a balance among convergence rate and global search effectiveness. This gradually decreases throughout the iterations before adjusting to a value that approximately corresponds to $visual_{min}$ and $step_{min}$, this makes them unable of conducting any additional local searches. Meanwhile, the third theoretical implication is to solve discrete machine scheduling models by transforming continuous vector solutions into discrete solution space.

An example is provided to explain the calculation of the proposed modified AFSA. Suppose AF represents a solution to the problems that will be solved at AFSA. In the case of UPMSP, the AF represents the solution sequence for the given problem. That is, the solution for AF is to randomly assign jobs to machines based on the number of jobs and

machines involved. Table 4.1 summarizes the solution structure for each of the five AF, assuming five jobs and two machines.

Table 4.1

Structure of the solution artificial fish

AF 1	Job 1	Job 2	Job 3	Job 4	Job 5
	M_1	M_2	M_1	M_2	M_1
AF 2	Job 1	Job 2	Job 3	Job 4	Job 5
	M_2	M_1	M_2	M_1	M_2
AF 3	Job 1	Job 2	Job 3	Job 4	Job 5
	M_1	M_1	M_2	M_2	M_1
AF 4	Job 1	Job 2	Job 3	Job 4	Job 5
	M_2	M_2	M_1	M_1	M_2
AF 5	Job 1	Job 2	Job 3	Job 4	Job 5
	M_1	M_2	M_2	M_1	M_2

Table 4.2 presents the processing times and due dates for these five jobs:

Table 4.2

The processing times and due dates

Jobs	**Processing times in M_1 (minutes)**	**Processing times in M_2 (minutes)**	**Due dates (minutes)**
Job 1	3	2	4
Job 2	4	1	3
Job 3	2	3	5
Job 4	5	2	6
Job 5	3	4	7

The following steps must be followed to solve this example using modified AFSA.

Step 1:

1. Initialize Parameters:
   - Number of AFs: 5
   - *Visual*: 2
   - *Step*: 1
   - Maximum iterations: 10

2. Generate Initial Population: this involves randomly assigning jobs to machines, as illustrated in Table 4.1.

Step 2: Searching

All AFs have a continuous position generated by the standard AFSA. Assume the continuous positions are as follows:

- **AF 1:** [0.23, 0.43, 0.1, 0.35, 0.5]
- **AF 2:** [0.5, 0.2, 0.7, 0.1, 0.3]
- **AF 3:** [0.15, 0.35, 0.25, 0.4, 0.6]
- **AF 4:** [0.4, 0.6, 0.3, 0.2, 0.5]
- **AF 3:** [0.1, 0.5, 0.2, 0.3, 0.4]

Step 3: Transformation method for discrete solutions

Sorting the continuous values and mapping them to job indices will convert them into discrete job assignments.

1. AF 1:
   - Continuous solution:

Job 1	Job 2	Job 3	Job 4	Job 5
M1	M2	M1	M2	M1
0.23	0.43	0.1	0.35	0.5

   - Sorted values:

Job 3	Job 1	Job 4	Job 2	Job 5
M1	M1	M2	M2	M1
0.1	0.23	0.35	0.43	0.5

   - Discrete job assignments: Job 3- Job 1- Job 5 on M1, Job 4 - Job 2 on M2

2. AF 2:

- Continuous solution:

Job 1	Job 2	Job 3	Job 4	Job 5
M2	M1	M2	M1	M2
0.5	0.2	0.7	0.1	0.3

- Sorted values:

Job 4	Job 2	Job 5	Job 1	Job 3
M1	M1	M2	M2	M2
0.1	0.2	0.3	0.5	0.7

- Discrete job assignments: Job 4 - Job 2 on M1, Job 5- Job 1- Job 3 on M2

3. AF 3:

- Continuous solution:

Job 1	Job 2	Job 3	Job 4	Job 5
M1	M1	M2	M2	M1
0.15	0.35	0.25	0.4	0.6

- Sorted values:

Job 1	Job 3	Job 2	Job 4	Job 5
M1	M2	M1	M2	M1
0.15	0.25	0.35	0.4	0.6

- Discrete job assignments: Job 1 - Job 2- Job 5 on M1, Job 3- Job 4 on M2

4. AF 4:

- Continuous solution:

Job 1	Job 2	Job 3	Job 4	Job 5
M2	M2	M1	M2	M1
0.4	0.6	0.3	0.2	0.5

- Sorted values:

Job 4	Job 3	Job 1	Job 5	Job 2
M2	M1	M2	M1	M2
0.2	0.3	0.4	0.5	0.6

- Discrete job assignments: Job 3- Job 5 on M1, Job 4 - Job 1- Job 2 on M2

5. AF 5:

- Continuous solution:

Job 1	Job 2	Job 3	Job 4	Job 5
M1	M2	M2	M1	M2
0.1	0.5	0.2	0.3	0.4

- Sorted values:

Job 1	Job 3	Job 4	Job 5	Job 2
M1	M2	M1	M2	M2
0.1	0.2	0.3	0.4	0.5

- Discrete job assignments: Job 1- Job 4 on M1, Job 3 - Job 5- Job 2 on M2

Step 4: Calculate the fitness function:

Compute the sum of makespan and total tardiness for each fish. Thus, the fitness values for each AF sequence of job assignments are as follows:

- Job Assignment of AF 1:

$$\left.\begin{array}{l} Completion\ time\ M1: Job\ 3 + Job\ 1 + Job\ 5 = 8 \\ Completion\ M2: \quad job\ 4 + job\ 2 \qquad\qquad = 3 \end{array}\right\} \rightarrow Makespan = 8$$

and $total\ tardiness = 2, then\ fitness\ value = 10$

- Job Assignment of AF 2:

$$\left.\begin{array}{l} Completion\ time\ M1: job\ 4 + job2 = 9 \\ Completion\ M2: \quad job\ 5 + job\ 1 + job\ 3 = 9 \end{array}\right\} \rightarrow Makespan = 9$$

and $total\ tardiness = 12, then\ fitness\ value = 21$

- Job Assignment of AF 3:

$$\left.\begin{array}{l} Completion\ time\ M1: job1 + job\ 2 + job5 = 10 \\ Completion\ M2: \ job\ 3 + \text{job}\ 4 = \qquad 5 \end{array}\right\} \rightarrow Makespan = 10$$

and $total\ tardiness = 7, then\ fitness\ value = 17$

- Job Assignment of AF 4:

$$\left.\begin{array}{l} Completion\ time\ M1: job\ 3 + job\ 5 = 5 \\ Completion\ M2: \ job\ 4 + job\ 1 + job\ 2 = 5 \end{array}\right\} \rightarrow Makespan = 5$$

and $total\ tardiness = 2, then\ fitness\ value = 7$

- Job Assignment of AF 5:

$$\left.\begin{array}{l} Completion\ time\ M1: job\ 1 + job\ 4 = 8 \\ Completion\ M2: \ job\ 3 + job\ 5 + job\ 2 = 8 \end{array}\right\} \rightarrow Makespan = 8$$

and $total\ tardiness = 7, then\ fitness\ value = 15$

Step 5: Stores the best solution found

$$(\text{AF } 1, \text{AF } 4, \text{AF } 5)$$

Step 6: Behavioral Procedures:

i. Prey behavior: Every AF looks for a better position within its visual range. If a better position is found, the AF moves towards it, such as:

- AF 1: Finds a position with a fitness value of 7 (the location of AF 4) by searching within its visual range. It advances to this position.
- AF 2: Conducts a search and identifies a position with a fitness value of 7 (the position of AF 4). It goes on toward this position.
- AF 3: Searches but fails to identify a better position within its visual range.
- AF 4: Currently has the best position with a fitness value of 7. It remains in its current position.
- AF 5: Searches and discovers a position with a fitness value of 3 (AF 4's position). This is the position it moves toward.

ii. Swarm behavior: If the swarm's center has a better fitness value, AF will move towards it, such as:

- The swarm center is the average position of all AFs, with a fitness value of 5.
- AF 3: It goes toward the center to possibly improve its position.
- AF 4: It has the best position, with a fitness value of 3. It remains in its current position.

iii. Follow behavior: AF follows the best-performing fish within their visual range.

- AF 3: identifies AF 4, which has a fitness value of 7, within its visual range. AF 3 goes to the position of AF 4.
- AF 4: It has the best position, with a fitness value of 7. It remains in its current position.

iv. Random behavior: The AF moves randomly if there is no better position within the visual range.

- AF 1: Moves randomly due to the inability to identify a better position within visual distance.
- AF 2: Moves randomly due to the inability to identify a better position within visual distance.

v. Aspiration behavior: AFs move in the direction of the best solution thus far.

- AF 1 discovers that AF 4 demonstrates a better fitness function. It moves to the position of AF 4.
- AF 2 also notes that AF 4 is more fit and aspires to adopt its position.
- AF 3 determines that AF 4 exhibits better fitness and looks to follow it.
- AF 5 discovers that AF 4 is more fit and is looking to take steps towards it.

Step 7: Improved Parameters:

- During the search process, adaptive visual and step parameters are implemented and dynamically adjusted.

Step 8: Update positions:

Reposition the fish according to their behaviors, such as:

- AF 1: moves towards the position of AF 4.
- AF 2: moves towards the position of AF 4.
- AF 3: moves towards the position of AF 4.
- AF 4: remains in its current position.
- AF 5: moves towards the position of AF 4.

Step 9: Check stopping criteria:

The algorithm should be stopped if the maximum number of iterations is reached or the solution converges. If not, perform the behavioral procedures again.

Step 10: End

The fish swarm's best position represents the best solution to the UPMSP. Hence, after executing the algorithm, it could identify the best solution as AF 4 with job assignment of Job 3 - Job 5 on M1, Job 4 - Job 1- Job 2 on M2, which have fitness value 7.

The computational cost to solve the discrete fuzzy multi-objective UPMSP increases significantly as the problem size increases because of the combinatorial complexity of the scheduling environment. Large-size problems require more memory and time. Nevertheless, the modified AFSA helps avoid this by improving both convergence speed and solution quality.

4.3 Hybrid Algorithm of Modified Artificial Fish Swarm Algorithm and Neighborhood Search Algorithm

Hybridization occurs when two or more algorithms are combined to improve a specific performance metric (Masrom et al., 2012). Notably, algorithms can be combined partially or completely to achieve the best features of the hybridization algorithm. The hybridization of various search methods, integrating population-based and single-solution search algorithms, has been extensively employed to address optimization problems (Wan et al., 2013). In particular, hybrid metaheuristic algorithms have become popular for Multi-Objective Optimization (MOO), producing the best outcomes for both practical and academic problems (Talbi, 2015). Meanwhile, population-based algorithms start with multiple solutions and search for a larger area of the search space, combining current solutions to generate new ones. Meanwhile, neighborhood search-based algorithms typically start with a random initial solution and attempt to improve iteratively using a fitness function. As a result, population-based solutions are less precise than neighborhood search algorithms. However, neighbourhood search algorithms are widely used because they solve optimization problems by iteratively executing the best possible move. Rather than spending more time in regions with less attractive solutions, neighborhood search algorithms investigate regions with reasonable solutions. Furthermore, the neighborhood search algorithm components make the algorithm attractive for problems with many constraints (Chen et al., 2022). Therefore, in this section, we utilized a neighborhood search algorithm to hybridize the proposed modified AFSA algorithm to solve UPMSP.

4.3.1 Neighborhood Search Algorithm Implementation

Neighborhood search was developed as an alternative algorithm to handle combinatorial optimization problems, mostly NP-hard, in various domains. This includes scheduling, assignment, computer channel balancing, vehicle routing, and cluster analysis, and others (Pirim et al., 2008). The essential elements of the neighborhood search algorithm are as follows:

1. **Initial Solution:**

In this research, the initial solution in the neighborhood search is based on the best solution discovered by the proposed modified algorithm. Note that the quality of the initial solution impacts the performance of the proposed hybrid algorithm.

2. **Neighborhood Structure**

A neighborhood structure is a set of operations that generates a new set of neighbor solutions by simply modifying a current solution. In this research, all neighborhoods of the current solution can be obtained by three operations: swap, reversion, and insertion between the locations of two random jobs.

3. **Evaluation**

Evaluate the quality of the neighborhood's solutions using the proposed multi-objective function.

4. **Move**

A modification to a solution is to move to the best solution in the neighborhood if it improves the objective function.

5. **Termination criterion**

The search may never end if the optimal value of the given problem is not known in advance. In practical terms, the search must end. In this research, a certain number of iterations is used as a stopping criterion.

4.3.2 The Proposed Hybrid Algorithm

To improve the proposed modified AFSA and achieve more precise solutions, a hybrid algorithm is created by integrating the proposed modified AFSA with the neighborhood search algorithm. The foundational framework for this hybridization is provided by the proposed modified AFSA algorithm, which explores the best possible solution thus far and implements it as an initial solution of the neighborhood search algorithm. To exploit the best solution in the search space, a neighborhood search algorithm can be easily integrated by generating a set of neighborhoods using three neighborhood operations. In the subsequent iterations, the operations are repeatedly performed from the given iterations until no further improvement can be achieved.

In order to fully exploit the search space and identify the best solutions, this research utilizes three neighborhood operations to enhance the intensity in a promising region, enhancing algorithm performance. These operations improve neighboring solutions by iteratively changing the current one. A neighborhood is essentially a set of solutions that can be obtained from the current one using three operations: swap, reversion, and insertion between the locations of two random jobs in the same solution. These operations are displayed in Figure 4.5, which adopted from Li and Yin (2013), who attempted to solve the multi-objective Flow Shop Scheduling Problem (FSSP) by presenting a new hybrid

algorithm combining multiple neighborhood search mechanisms with the cuckoo search algorithm to improve population diversity. This includes enhancing the quality of the solution, boost the ability to exploit searches, and obtaining a better solution in the search space. The following is a description of the neighborhood operations:

- **Swap operation:** A solution is created by randomly swapping two different jobs assigned to the same machine.
- **Reversion operation:** Reverse the subsequence between two randomly selected jobs assigned on different machines from the current sequence.
- **Insertion operation:** Select two different job positions at random from the current solution and put the back position before the front. The resulting sequence represents a neighbor. If the index of the first job is less than the second one, insert the job at this index into the position after the index job at the second one. If not, the job at the first index is inserted prior to the second one.

The neighborhood operations are performed on the selected solution to exploit the local neighborhood and improve the current solution, which is then taken and replaced with the new one. Otherwise, if a current solution cannot be improved after a predetermined number of iterations, the algorithm stops and considers the final local optimum solution better than the hybrid one.

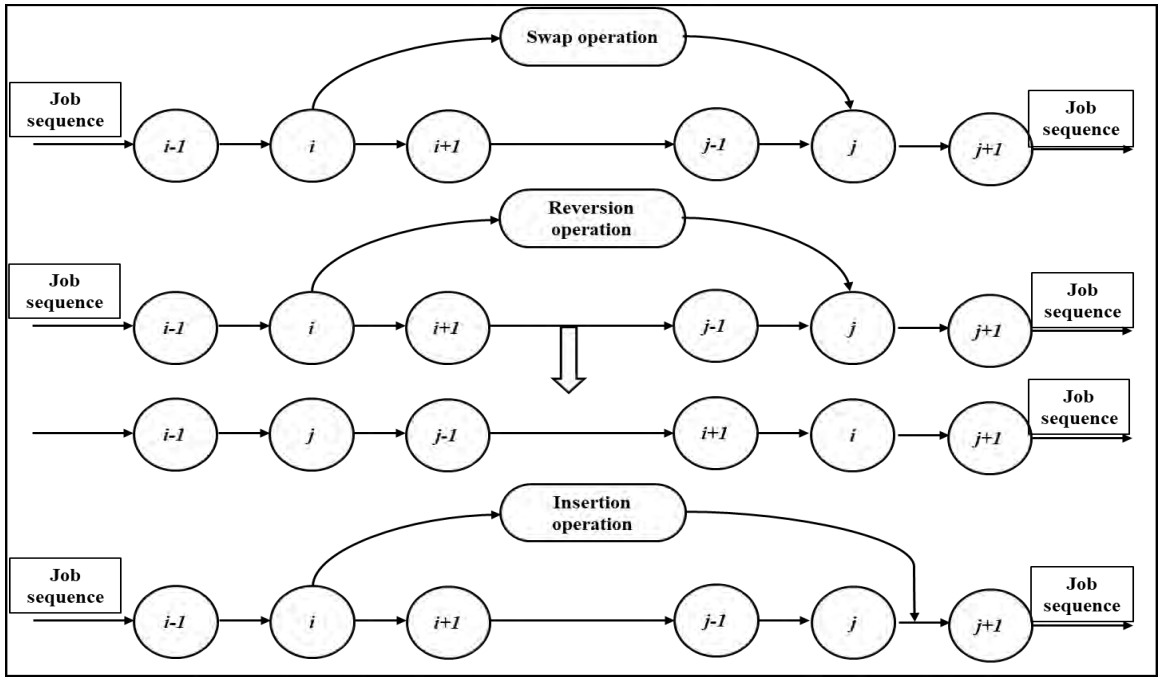


Figure 4.5. The neighborhood operations.

The detailed steps of the implementation process for the neighborhood search algorithm can be summarized as follows:

1. Forming an initial solution in the neighborhood search based on the best solution found by the proposed modified algorithm.
2. Calculating the value of the proposed multi-objective function.
3. Three neighborhood search operations: 1. Swap; 2. Reversion; 3. Insertion.
4. Evaluation of the best neighbor: After determining the best neighbor, it is compared with the current solution. If that leads to a better-than-current solution, the best neighbor considers the new solution, and the new solution is considered the current solution. Otherwise, the current solution will remain.
5. Repeat steps 1 to 4 to update the Best New Solution. Each iteration includes executing neighborhood search operations and updating the best solution in a candidate list.

6. After a certain number of iterations with no improvement in the multi-objective value, the algorithm terminates, and the best solution is presented as the optimal solution. Otherwise, the list will be updated, and operations will repeat.

Let us apply neighborhood operations (swap, reversion, and insertion) to the best solution AF 4, obtained from the previous example with sequence: Job 3 - Job 5 on M1, Job 4 - Job 1 - Job 2 on M2, and a fitness value of 7 which was solved by a modified algorithm in section 4.2.2.

Neighborhood Operations

1. **Swap Operation**

Swap two jobs in the sequence.

- **Original**: Job 3 - Job 5 on M1, Job 4 - Job 1 - Job 2 on M2
- **Swap Job 3 and Job 5**: Job 5 - Job 3 on M1, Job 4 - Job 1 - Job 2 on M2

$$\left.\begin{array}{c} Completion\ time\ M1: job\ 5 + job\ 3 = 5 \\ Completion\ M2:\ job\ 4 + job\ 1 + job\ 2 = 5 \end{array}\right\} \rightarrow Makespan = 5$$

and $total\ tardiness = 2, then\ fitness\ value = 7$

2. **Reversion Operation**

Reverse the order of jobs in the sequence.

- **Original**: Job 3 - Job 5 on M1, Job 4 - Job 1 - Job 2 on M2
- **Reverse:** Job 5 - Job 3 on M1, Job 2 - Job 1- Job 4 on M2

$$\left.\begin{array}{c} Completion\ time\ M1: job\ 5 + job\ 3 = 5 \\ Completion\ M2:\ job\ 2 + job\ 1 + job\ 4 = 5 \end{array}\right\} \rightarrow Makespan = 5$$

and $total\ tardiness = 0, then\ fitness\ value = 5$

3. **Insertion Operation**

   Insert a job into a different position in the sequence

- **Original**: Job 3 -Job 5 on M1, Job 4 - Job 1 - Job 2 on M2
- **Insert Job 3 between Job 4 and Job 1**: Job 5 on M1, Job 4 - Job 3 - Job 1 - Job 2 on M2

$$\left.\begin{array}{c}\text{Completion } time\ M1\text{: } job\ 5 = 3 \\ \text{Completion } M2\text{: } job\ 4 + job\ 3 + job\ 1 + job2 = 8\end{array}\right\} \rightarrow Makespan = 8$$

$$\text{and } total\ tardiness = 8, then\ fitness\ value\ = 16$$

Accordingly, after applying neighborhood operations, the best sequence comes from the reversion operation, with sequence Job 5 - Job 3 on M1, Job 2 - Job 1- Job 4 on M2, and fitness value 5.

4.3.2.1 Solution Representation

Solution representation is a significant aspect of the development of metaheuristic algorithms. The discrete encoding scheme of the neighborhood search algorithm cannot be directly addressed by the continuous encoding scheme of the four main behaviors of AFSA in the hybrid algorithm. Among the most difficult tasks is forming a direct relationship between the job sequence and the vector of individuals in the AFSA. This ensures that the better neighborhood solution produced by the neighborhood search algorithm can be applied to the AFSA.

Since the best solution identified by the proposed modified algorithm is a discrete solution, and the neighborhood search algorithm is a discrete optimization algorithm, the better solution in the neighborhood search space is also discrete. Meanwhile, since this algorithm

is a hybrid, the improved output solution must be returned to the AFSA population to ensure that the entire population of AFSA is improved. When the hybrid algorithm returns to the next iteration, it can benefit from better solutions improved by the neighborhood search algorithm. Hence, we should convert the better neighborhood solution from a discrete solution to a continuous solution to deal with continuous equations of the four main behaviors of standard AFSA. Thus, to transform the discrete job sequence into a continuous individual vector in this subsection, we propose encoding a discrete solution with integer numbers using a random key. Subsequently, the continuous solution is decoded using these values as sort keys. The value in the first order of the jobs sequence is selected as a ranked value representation in the continuous vector in AFSA, and it is supposed to be equal to a random number between 0.1 and 0.2. Correspondingly, the value in the second order of the jobs sequence is picked as a ranked value representation in the continuous vector in AFSA and is equal to the previous value of the continuous vector plus a random number between 0.1 and 0.2. Accordingly, all the values of the job sequence will be managed to convert the job sequence to a continuous vector.

We now demonstrate the proposed transformation method with a simple example in Figure 4.6 Let $S = [4\ \ 1\ \ 3\ \ 5\ \ 2]$ represent the better neighborhood solution of the neighborhood search algorithm, $E = 0.1$ is a random variable $\in [0.1\ 0.2]$, and A represents a continuous solution. Let the solution be indexed by integer numbers $\mathrm{N} = [1\ 2\ 3\ 4\ 5]$. After a combination of solution and index, we obtain:

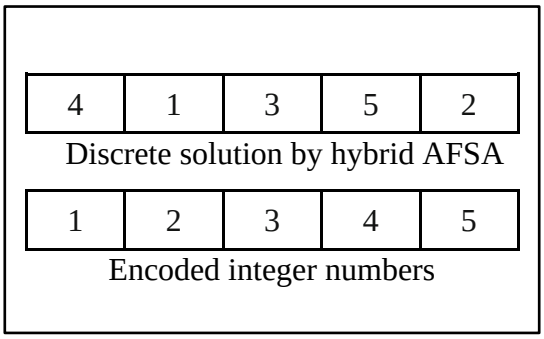


Figure 4.6. An example of the solution representation in hybrid AFSA

$A(S(1)) = E;\ A(4) = 0.1;$

$A\big(S(2)\big) = A(1) = A(4)$ +random variable $\in [0.10.2]$.

- Let us assume the random variable is 0.15: $(A(1) = 0.1 + 0.15 = 0.25)$

$A\big(S(3)\big) = A(3) = A(1)$ +random variable $\in$ [0.1 0.2].

- Let us assume the random variable is 0.12: $(A(3) = 0.25 + 0.12 = 0.37)$

  $A(S(4)) = A(5) = A(3)$ +random variable $\in$ [0.1 0.2].

- Let us assume the random variable is 0.18: $(A(5) = 0.37 + 0.18 = 0.55)$

$A(S(5)) = A(2) = A(5)$ +random variable $\in$ [0.1 0.2].

- Let us assume the random variable is 0.13: $(A(2) = 0.55 + 0.13 = 0.68)$

Resulting solution: $A = [0.25, 0.68, 0.37, 0.1, 0.55]$.

The output solution A is normalized by dividing it by its norm. Thus, $A = [0.27, 0.7, 0.37, 0.14, 0.52]$ is the final normalized result of the solution, which corresponds to the final continuous solution combined in AFSA.

4.3.2.2 The Flowchart of Proposed Hybrid AFSA

The flowchart of the proposed hybrid AFSA algorithm is illustrated in the brown part of Figure 4.7, which details each step involved in the hybrid algorithm. In essence, applying the neighborhood search algorithm to the better solution discovered in the modified AFSA algorithm was the first step in the hybrid part.

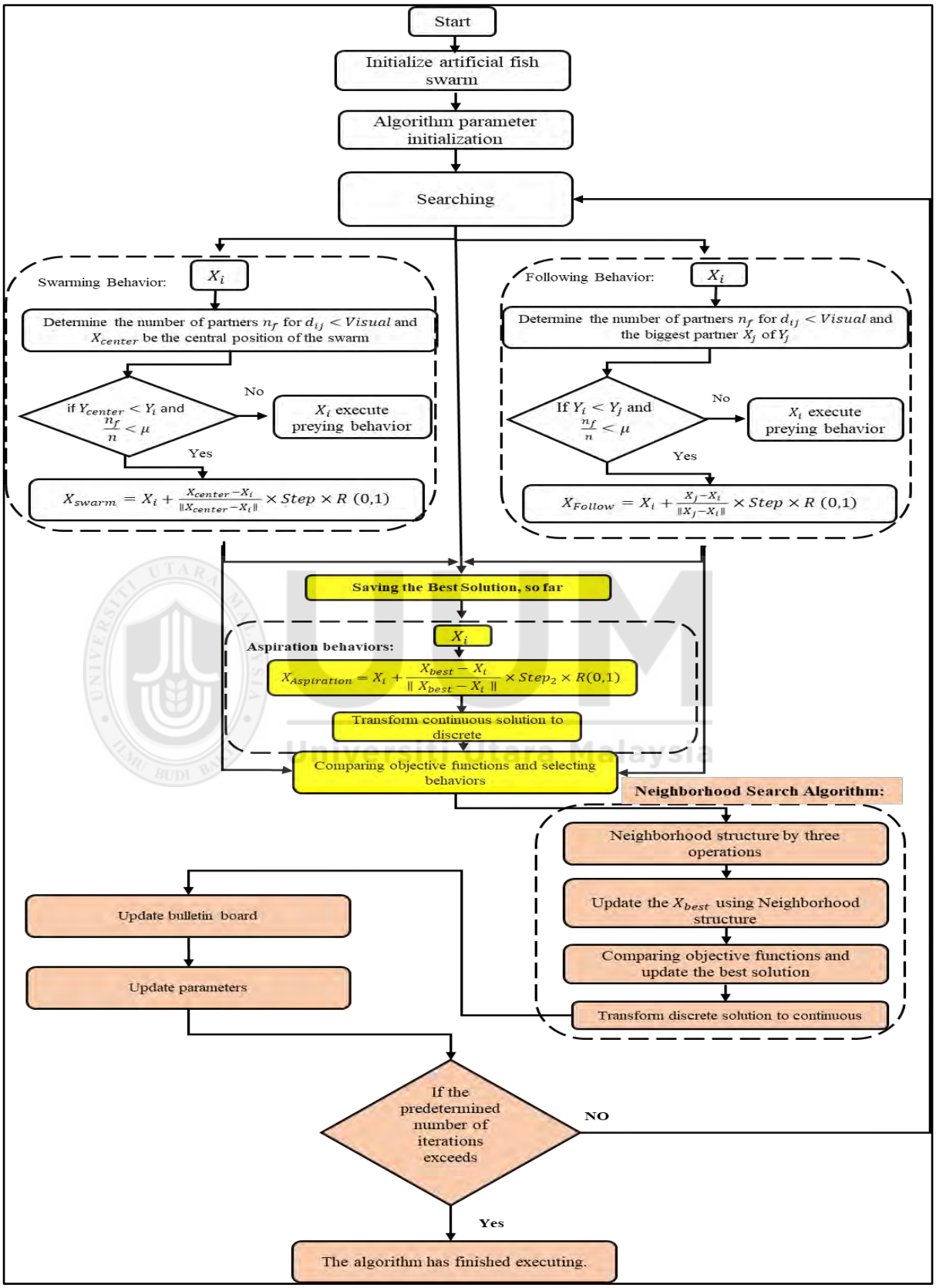


Figure 4.7. Flowchart for the proposed hybrid AFSA algorithm

Three neighborhood search operations were searched in the search space: swap, reversion, and insertion to determine the best neighbor of the discovered better solution. Subsequently, the best neighbor was evaluated by comparing its objective function with the current solution. Accordingly, the best neighbor will update the bulletin board and consider the new solution if proven superior to the current one. This hybrid method produces considerably superior results compared to the modified AFSA algorithm, leading to more accurate results.

The hybrid algorithm will overcome the computational bottlenecks that occur from large-size problems by focusing the search effort on promising solutions instead of trying to explore the entire vast search space. The hybrid algorithm is specifically designed by employing three neighborhood search operations, the AFSA framework provides the global exploration needed to move between different regions of the search space, preventing premature convergence to poor local. The integrated neighborhood search then performs the intensive local exploitation required to rapidly identify the best solution within each promising region. This combination allows the algorithm to effectively balance the search while also showing significant improvements in convergence speed compared to standard AFSA, resulting in higher-quality solutions than either a pure exploration or exploitation strategy could achieve alone.

4.4 Chapter Summary

This chapter focuses on providing a detailed explanation of the proposed algorithms used to solve the proposed multi-objective UPMSP. This chapter is presented in two phases that outline the fundamental framework of our research. The first phase involved developing

a modified algorithm for the AFSA. We modified the algorithm in three distinct aspects. The first aspect adds a new behavior called aspiration behavior to the four main AFSA behaviors to overcome AFSA's drawbacks and make it more appropriate for the proposed problem. Meanwhile, the second aspect of modification to our algorithm was the adoption of improved parameters of step and visual to achieve a balance between convergence rate and global search capability. Additionally, the third aspect is to adapt the AFSA algorithm in solving discrete UPMSPs and improve the original algorithm's search capabilities. Therefore, the main contribution of modified AFSA is the use of transformation method to convert the continuous solution into a discrete solution. Building on that, the second phase is to develop a hybrid algorithm that combines the proposed modified algorithm with a neighborhood search algorithm based on three neighborhood operations. While, all of these are considered as theoretical contributions to the development of the proposed modified and hybrid AFSA algorithms; therefore, the practical contribution is to apply them to the randomly generated computational experiment sets to assess the algorithm's performance. This can boost UPMSP performance and productivity, thereby improving scheduling quality across various industries.

CHAPTER FIVE

RESULTS AND DISCUSSION

5.1 Introduction

This chapter introduces the computational results of the proposed modified and hybrid Artificial Fish Swarm Algorithm (AFSA) algorithms to achieve the objectives of this study. The discussion starts with Section 5.2, which discusses the computational design of the proposed modified algorithm. Meanwhile, Section 5.3 discusses the computational results of implementing a hybrid AFSA, where the comparison has been conducted with various algorithms. This includes the standard AFSA, proposed modified algorithms, and metaheuristic algorithms. Finally, Section 5.4 presents a summary of this chapter.

5.2 The Computational Design of the Proposed Modified Algorithm

This subsection explains the computational design that was conducted to evaluate the performance of the modified algorithm in comparison with standard AFSA and other modified algorithms. The algorithm performance was assessed using random test sets of small, medium, and large sizes to analyze the problem, as inspired by (Lin & Lin, 2015). The following problems have ten instances for each size and are executed ten times: a small-sized problem with N = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50] and M = [3, 3, 4, 4, 5, 5, 6, 6, 7, 7]; a medium-sized problem with N = [60, 80, 100, 120, 140, 160, 180, 200, 220, 240] and M = [8, 8, 9, 9, 10, 10, 11, 11, 12, 12]; and a large-sized problem with N = [250, 270, 290, 310, 330, 350, 370, 390, 410, 430] and M = [13, 13, 14, 14, 15, 15, 16, 16, 17, 17]. All runs are executed on a personal computer with 12th Gen Intel(R) Core (TM) i7-1280P 2.00 GHz and 16 GB RAM under Windows 11 (64bit) using MATLAB R2023a. Table 5.1 describes the different parameter settings on the algorithm.

Table 5.1

Parameter settings for the proposed modified algorithm

Parameter	**Values**
Try-number	10
Fish population ($nPop$)	40
Visual	30
$visual_{min}$	30
$Step_1$	1
$Step_2$	2
$step_{min}1$	1
$step_{min}2$	2
Crowd factor (μ)	0.3
Arbitrary value (σ) ($0.5 < \sigma < 1$)	0.6
Maximum iteration (max-iterations)	1000

The experiments were conducted to fine-tune parameters to determine the best values of the following parameters to be used by the proposed modified algorithm before its comparison. This is achieved by repeatedly running each parameter set; hence, all parameters must undergo many iterations until the combination that yields the best objective function value is found.

Table 5.1 signifies that the number of attempts in preying behavior, referred to as try-number, is 10. The crowd factor ($0 < \mu < 1$) is 0.3. Meanwhile, the fish population (*nPop*) is 40, maximum iterations (max-iterations) is 1000, a fixed maximum iteration value has been set for all problem sizes to ensure that the algorithm ends within a maximum number of steps, regardless of the problem's complexity or size. At the same time, an arbitrary value ($\sigma = 0.6$), which must be within the range $0.5 < \sigma < 1$. $Step_1$ and $step_{min}1 = 1$; $Step_2$

and $Step_{min}$ = 2. $Visual$ and $visual_{min}$ = 30, are the best values across all evaluated performances.

5.3 The Computational Results of the Proposed Modified Algorithm

The proposed modified algorithm has been implemented and executed successfully, and the resulting data has been generated. Thus, in this subsection, the computational results of the proposed modified algorithm are illustrated for small, medium, and large-sized problems in Tables 5.2, 5.3, and 5.4, respectively. The results of the proposed modified algorithm are presented together with those obtained from standard AFSA, Modified 1, Modified 2, and the three methods in Modified 3. The results of these algorithms are compared to assess the performance of the proposed algorithm. The grey color represents the best results among the comparative algorithms for each problem.

Table 5.2. signifies the results of small-sized problem instances, it is notable that the proposed algorithm exhibits relative performance improvements when dealing with small-sized problems. Furthermore, it is noteworthy that the best minimum values that the proposed algorithm yields are 23.25, 42, and 47.25, which all occurred in the instances of (N = 5, M = 3), (N = 10, M = 3), and (N = 15, M = 4), respectively. Specifically, Modified 3-Method 3 obtains the best results compared with other algorithms in the next five instances from (N = 20, M = 4) to (N = 40, M = 6). Conversely, AFSA obtained the best results in the last two instances (N = 45, M = 7) and (N = 50, M = 7).

In all tests, mean and STD measures are used to assess the reliability and stability of the proposed algorithm's performance on various set sizes, as well as to compare it to other algorithms. A smaller mean value represents the best average solution executed by the

algorithms, while a lower STD signifies that the results are closer to the mean, implying more consistent or stable performance. Table 5.2 indicates that the proposed algorithm produced the best mean values in the first three instances, with values of 23.475, 43.275, and 51.975, respectively. Additionally, the algorithm achieved the best STD value in six instances: 0.711512, 1.589025, 1.852363, 1.371587, 2.243045, and 1.864135, for (N = 5, M = 3), (N = *20*, M = 4), (N = *25*, M = 5), (N = *30*, M = 5), (N = *40*, M = 5), and (N = 50, M = 6), respectively, for small-sized problems.

Tables 5.3 and 5.4 summarize the results of medium and large-sized problems. However, as the number of jobs and machines increases to medium and large sizes, the proposed algorithm demonstrates an even more pronounced enhancement in its superior performance.

Table 5.2

The computational results for the proposed modified algorithm for small-sized problems

	N	M		Proposed modified	AFSA	Modified 1	Modified 2	Modified 3 - method 1	Modified 3 - method 2	Modified 3 - method 3
1	**5**	**3**	**Mean**	23.47	23.47	23.70	24.60	23.92	24.07	23.70
			STD	0.71151	0.71151	0.94868	1.16189	1.08685	1.38969	0.94868
			Min	23.25	23.25	23.25	23.25	23.25	23.25	23.25
			Max	25.50	25.50	25.50	25.50	25.50	27.00	25.50
2	**10**	**3**	**Mean**	43.27	43.80	44.55	43.95	44.55	44.47	43.57
			STD	1.32523	0.88034	2.12720	1.66583	2.156385	2.06306	1.02774
			Min	42.00	42.75	42.00	42.00	42.00	42.00	42.00
			Max	45.75	45.75	47.25	47.25	48.75	47.25	45.00
3	**15**	**4**	**Mean**	51.97	53.62	53.17	52.12	54.82	56.02	51.99
			STD	2.64693	3.14742	2.67978	2.43027	4.09954	2.69374	2.97489
			Min	47.25	48.00	50.25	48.00	49.50	51.00	48.75
			Max	55.50	57.75	57.75	54.75	62.25	59.25	57.00
4	**20**	**4**	**Mean**	64.05	62.55	66.30	62.77	66.37	66.90	61.95
			STD	1.58902	1.62788	2.94108	2.26522	2.65165	1.65075	1.73925
			Min	62.25	60.00	62.25	59.25	62.25	63.75	58.50
			Max	66.75	65.25	73.50	66.00	69.75	69.00	64.50
5	**25**	**5**	**Mean**	67.08	65.33	70.28	67.88	70.03	72.08	65.30
			STD	1.85236	1.98273	2.84421	2.43027	3.06741	4.61586	1.97132
			Min	63.75	63.00	65.25	64.50	66.75	64.50	62.25
			Max	69.25	70.00	73.75	72.25	76.00	78.00	70.00
6	**30**	**5**	**Mean**	76.28	73.65	79.28	74.25	78.98	80.63	72.60
			STD	1.37158	1.57321	3.85762	2.23606	2.39921	2.37828	1.68819
			Min	74.25	72.00	75.00	70.50	76.50	78.75	70.50
			Max	78.00	75.75	87.75	77.25	84.00	86.25	75.00
7	**35**	**6**	**Mean**	87.90	85.13	91.65	83.55	90.85	97.35	83.33
			STD	3.27490	2.37828	7.58122	3.28253	4.74517	4.79322	2.67978
			Min	83.25	81.75	82.50	78.75	86.25	91.50	78.75

			Max	93.75	88.50	106.50	88.50	99.75	105.75	87.00
8	**40**	**6**	**Mean**	98.88	98.50	108.55	97.75	105.55	109.38	98.00
			STD	2.24304	3.39116	9.38216	3.22102	5.61471	11.39764	3.61132
			Min	96.25	94.00	102.25	94.00	96.25	100.00	92.50
			Max	102.25	104.50	133.75	104.50	116.50	132.25	103.75
9	**45**	**7**	**Mean**	95.03	94.73	105.03	95.85	107.03	116.80	96.45
			STD	4.18404	3.70894	8.32453	3.93382	7.31574	11.54267	4.86312
			Min	90.00	87.75	96.00	88.50	94.50	102.00	89.25
			Max	105.00	102.00	120.75	100.50	117.25	137.25	107.25
10	**50**	**7**	**Mean**	94.65	93.60	108.65	93.13	107.48	112.95	96.38
			STD	1.86413	2.20227	7.22284	2.11886	4.89961	9.46470	3.10745
			Min	91.50	89.25	97.50	90.00	98.25	99.00	91.50
			Max	97.50	96.75	123.00	97.25	114.75	130.50	100.50

Table 5.3

The computational results for the proposed modified algorithm for medium-sized problems

	N	**M**		**Proposed modified**	**AFSA**	**Modified 1**	**Modified 2**	**Modified 3 -method 1**	**Modified 3 - method 2**	**Modified 3 - method 3**
1	**60**	**8**	**Mean**	65.90	64.65	69.00	65.45	69.80	72.35	66.45
			STD	1.44913	1.15590	2.47206	1.40336	1.70293	4.08282	1.32182
			Min	62.50	63.00	66.00	63.00	67.50	68.50	65.00
			Max	68.00	66.00	73.50	67.00	73.00	80.50	69.50
2	**80**	**8**	**Mean**	117.35	112.20	148.50	113.65	155.20	164.80	128.10
			STD	8.3335	6.028635	29.3342803	7.742559	25.2005291	19.3249982	12.8101522
			Min	107.50	101.00	118.50	102.50	123.00	133.00	110.00
			Max	130.00	121.50	214.50	124.50	201.50	197.50	153.50
3	**100**	**9**	**Mean**	110.00	112.80	133.90	113.90	147.10	141.30	121.25
			STD	3.85140	3.14642	12.84912	4.67736	24.41515	14.21501	9.98123
			Min	105.50	106.00	117.50	108.50	117.50	117.50	113.50

			Max	118.00	117.50	157.00	121.50	195.50	162.00	148.50
4	**120**	**9**	**Mean**	141.65	148.20	195.85	146.75	218.10	254.90	167.30
			STD	10.43245	5.86988	37.71313	18.29731	39.08949	65.58023	26.46297
			Min	123.50	141.50	147.50	131.50	173.00	164.00	136.00
			Max	160.00	158.50	263.00	189.50	277.50	378.00	211.50
5	**140**	**10**	**Mean**	151.75	156.15	231.25	153.90	238.20	297.30	177
			STD	18.22430	11.18046	28.75591	11.17487	33.30515	54.22238	13.01708
			Min	136.50	136.50	201.00	142.50	193.00	221.00	154.00
			Max	196.00	178.50	274.00	176.50	285.50	393.00	198.50
6	**160**	**10**	**Mean**	181.00	196.20	307.35	193.80	363.10	449.60	240.85
			STD	17.02449	17.20982	59.19602	9.72453	90.84076	105.06606	26.58429
			Min	161.50	173.50	230.50	178.50	221.50	348.00	192.00
			Max	215.00	227.50	447.50	212.50	547.00	678.00	298.00
7	**180**	**11**	**Mean**	175.15	177.95	299.65	181.45	326.70	456.30	217.45
			STD	12.71143	15.90606	70.88764	12.38604	46.19235	94.89766	23.23604
			Min	154.00	154.50	208.50	163.00	269.50	326.00	183.50
			Max	196.50	201.50	419.00	195.00	423.00	655.50	265.50
8	**200**	**11**	**Mean**	203.85	213.00	310.30	204.50	393.20	580.55	240.60
			STD	18.0909	20.45863	33.49643	11.19524	101.62873	99.48993	26.42158
			Min	181.50	192.50	269.50	187.50	282.50	372.00	209.50
			Max	238.00	248.00	364.00	218.00	581.00	753.50	290.00
9	**220**	**12**	**Mean**	281.10	317.20	752.55	302.90	837.35	1138.65	402.90
			STD	60.7096	50.07949	116.87539	27.08403	141.48813	306.00263	58.37845
			Min	205.00	238.00	538.00	262.50	660.50	697.50	340.00
			Max	398.00	402.5.00	915.00	345.50	1034.00	1672.50	558.00
10	**240**	**12**	**Mean**	324.25	373.45	944.45	376.15	931.85	1702.00	442.95
			STD	47.46768	47.62379	231.36508	52.06942	162.44384	418.95703	66.64685
			Min	251.50	308.50	621.00	294.50	664.50	1206.00	351.00
			Max	375.50	446.50	1319.50	457.00	1214.50	2392.50	553.50

Table 5.4

The computational results for the proposed modified algorithm for large-sized problem

	N	M		Proposed modified	AFSA	Modified 1	Modified 2	Modified 3 -method 1	Modified 3 -method 2	Modified 3 -method 3
1	**250**	**13**	**Mean**	344.60	377.95	918.30	356.85	956.35	1705.35	481.45
			STD	56.49916	45.5689	195.28187	67.45412	166.97904	448.824146	65.43930
			Min	262.00	287.50	629.00	287.00	684.00	1058.00	384.50
			Max	422.50	441.00	1165.50	488.00	1265.00	2419.50	585.00
2	**270**	**13**	**Mean**	336.15	430.40	1142.90	410.50	1181.60	2809.50	515.65
			STD	43.07361	78.93943	193.10371	47.35387	157.18191	954.20147	80.32020
			Min	270.00	276.00	715.00	347.00	971.50	1876.00	429.00
			Max	427.50	527.50	1462.00	499.00	1451.00	4390.50	654.00
3	**290**	**14**	**Mean**	573.90	634.95	1753.05	566.20	1790.50	3777.80	866.90
			STD	118.7923	76.78341	138.85473	90.48732	292.94785	681.76899	118.18107
			Min	409.00	452.50	1463.50	432.50	1249.50	2580.00	715.50
			Max	770.00	726.00	1947.00	697.00	2289.50	4790.00	1123.50
4	**310**	**14**	**Mean**	432.80	495.00	1440.40	448.30	1586.95	4748.05	608.55
			STD	80.23611	106.84310	180.93120	79.73156	247.05368	1065.97672	78.53677
			Min	303.00	404.00	1191.00	327.50	1313.00	3215.00	514.00
			Max	604.00	643.00	1799.00	576.50	1904.50	6634.00	721.50
5	**330**	**15**	**Mean**	637.45	674.55	2187.55	699.15	2123.25	6812.70	985.80
			STD	79.19822	142.63110	339.22210	124.84080	317.49219	1292.51091	178.97457
			Min	559.00	542.50	1695.50	448.50	1543.00	4593.00	716.50
			Max	806.50	1007.00	2685.00	890.00	2594.50	8800.00	1270.50
6	**350**	**15**	**Mean**	779.25	1068.15	2767.50	1060.10	3106.25	9727.05	1408.65
			STD	144.62310	131.53670	475.51866	148.34870	367.18586	804.67317	154.26385
			Min	565.00	898.00	2014.50	884.00	2624.50	8196.50	1121.00

			Max	1075.00	1310.00	3754.00	1354.00	3847.50	10681.50	1557.50
7	**370**	**16**	**Mean**	652.70	634.75	1965.45	689.65	2187.80	8790.50	927.10
			STD	133.58250	144.99350	257.42921	135.10530	309.22225	752.91385	208.92606
			Min	449.50	475.50	1510.00	463.50	1772.00	6986.00	668.00
			Max	915.50	918.50	2388.50	847.00	2764.00	9816.00	1223.50
8	**390**	**16**	**Mean**	993.90	1304.75	3357.75	1120.05	3575.90	11184.05	1705.40
			STD	117.98940	253.02750	637.85283	199.10580	375.70303	627.13074	325.74903
			Min	801.00	891.00	2647.50	847.00	3150.00	10149.50	1369.50
			Max	1187.50	1623.00	4380.50	1392.00	4202.00	12272.00	2289.00
9	**410**	**17**	**Mean**	866.35	1003.40	2927.10	983.95	2892.75	10599.05	1325.1
			STD	221.43150	201.19790	212.07019	207.73790	449.84066	611.26946	109.24406
			Min	517.00	649.00	2584.50	841.50	2437.50	9709.00	1191.00
			Max	1193.00	1354.50	3216.00	1496.00	3794.00	11811.50	1580.50
10	**430**	**17**	**Mean**	1178.25	1325.90	3723.55	1270.55	4417.70	12701.75	1798.85
			STD	269.32160	270.05050	541.57863	241.28110	561.60970	871.22099	266.45950
			Min	696.50	938.00	2752.00	977.00	3666.50	11626.50	1411.50
			Max	1671.00	1766.5.00	4553.00	1731.50	5297.00	14835.50	2269.50

It is observed that the proposed algorithm provides the best minimum values in nine instances of medium-sized problems: 62.5, 105.5, 123.5, 136.5, 161.5, 154, 181.5, 205, and 251.5. All occurred in the instance of (N = 60, M = 8), (N = 100, M = 9), (N = 120, M = 9), (N = 140, M = 10), (N = 160, M = 10), (N = 180, M = 11), (N = 200, M = 11), (N = 220, M = 12), and (N = 240, M = 12), respectively. Meanwhile, the standard AFSA provides the best minimum values in the second instance when (N = 80, M = 8). In large-sized problems, the proposed algorithm begins improving superiority performance and provides the best minimum values in nine instances, and offers the values 262, 270, 409, 303, 565, 449.5, 847, 517, and 696.5. All occurred in the instance of (N = 250, M = 13), (N = 270, M = 13), (N = 290, M = 14), (N = 310, M = 14), (N = 350, M = 15), (N = 370, M = 16), (N = 390, M = 16), (N = 410, M = 17), and (N = 430, M = 17), respectively. The Modified 2 provides the best minimum value in the fifth instance when (N = 330, M = 15).

The results reveal that, for medium and large-sized problems the proposed algorithm achieved the best mean values in eight instances, such as 110 and 336.15 for (N = *100*, M = 9) and (N = *27*0, M = 13), respectively. This includes the best STD values in four instances, such as 3.14642 and 79.19822 for (N = *100*, M = 9) and (N = *33*0, M = 15), respectively. This indicate that the superiority and efficiency of the proposed algorithm improve as the number of machines and jobs increases, and low values of STD indicate further constancy in the algorithm's performance. The best possible solutions for the proposed model are illustrated in the figures after ten runs of each algorithm. The proposed algorithm is guaranteed to converge to the best domain solutions for each three-size test problem.

The graph illustrates the proposed algorithm's performance in Figures 5.1, 5.2, and 5.3 on all size test problems. In all figures, the x-axis represents the number of iterations used in

the comparison process, while the y-axis represents the function evaluation in terms of getting closer to minimum values. As the number of jobs and iterations increases, the proposed algorithm has a notable impact when approaching the best values, clarifying the best convergence to the best solution. This demonstrates that the proposed algorithm can stably converge when compared to other algorithms.

It is evident that in almost all test instances, the minimum values of AFSA, Modified 1, Modified 2, Modified 3-Method 1, Modified 3-Method 2, and Modified 3-Method 3 algorithms are higher than our proposed modified AFSA. This also indicates that the proposed modified AFSA algorithm can obtain a better multi-objective function. Consequently, alongside illustrating the performance, it can also demonstrate the overall effectiveness and solution impact of the proposed modified AFSA compared to other algorithms across the three test sets, as depicted in Figures 5.1, 5.2, and 5.3. The proposed modified AFSA demonstrates that it is better than the six algorithms.

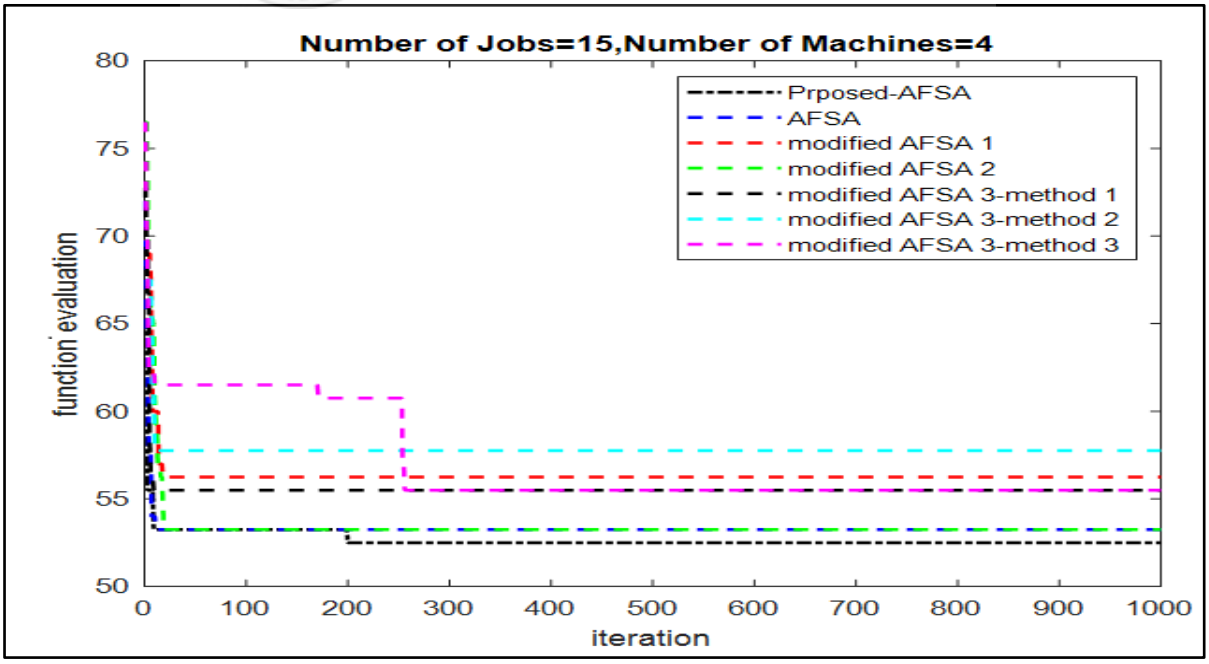


Figure 5.1. The performance of the proposed modified algorithm for sample small problem

Figure 5.1 illustrates the graph for sample small-sized problem with (N = 15, M = 4). This graph illustrates the effectiveness of the proposed modified algorithm and different AFSA versions in improving the quality of their solutions over a number of iterations. The convergence performance of various AFSA versions is represented by each colored and styled line. At iteration 0, the function evaluation of all algorithms is relatively high (between 65 and 75), which indicates that their solutions are not yet their best. In the first 50–100 iterations, almost all of algorithms show a significant decrease in function evaluation. In contrast to the other algorithms, the modified AFSA 3-method 3 (magenta line) shows a slower initial convergence and remains a higher function evaluation for a longer period of time before declining, and the path is much less stable, with a significant jump around iteration 250. At iteration 1000, the final solution reaches the lowest possible function evaluation. AFSA (blue dashed) and modified AFSA 2 (green dashed) converge to slightly higher values, around 54-55. Modified AFSA 1 (red dashed line) and modified AFSA 3-method 1 (solid black line) converge at around 56-57. The modified AFSA 3-method 2 (cyan dashed) performs the worst, with a value around 58-59. Proposed-AFSA (black dashed) appears to have the best (lowest) function evaluation, settling around 50-53, and demonstrates fast convergence speed by achieving the best value within the first 50-100 iterations.

This means that is the most effective in finding high quality solution for this specific instance. Figure 5.2 illustrates the graph for a sample medium-sized problem with (N = 220, M = 12), where the search space for best solutions is greatly expanded, making the improvement process significantly more difficult.

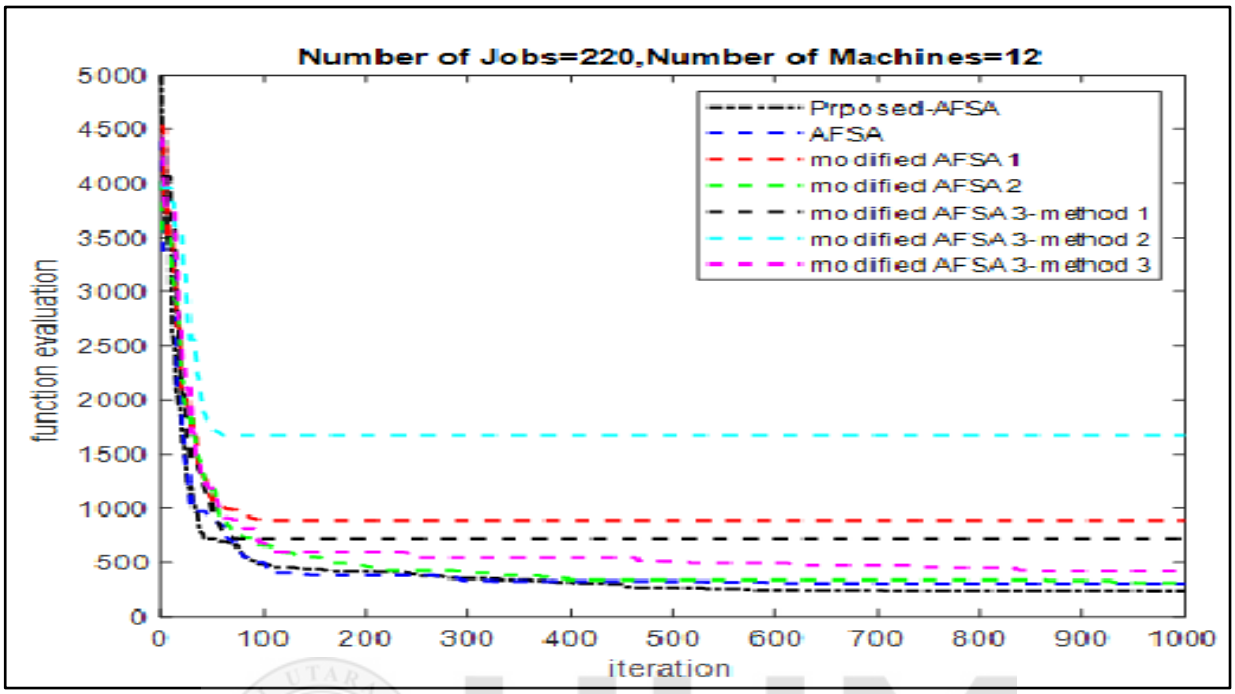


Figure 5.2. The performance of the proposed modified algorithm for sample medium problem

At iteration 0, all algorithms start with very high function evaluations (around 3500-4500), indicating the increased complexity of the problem. The initial randomly generated solutions are far from the best. All algorithms continue to show a rapid initial drop in function evaluation. In particular, both AFSA and Proposed-AFSA exhibit a very quick initial drop during the first 50–100 iterations, indicating fast initial speed convergence. Meanwhile, the proposed AFSA clearly reaches its best solutions around iteration 600 and remains stable until iteration 1000. This indicates that they are rapidly improving their initial solutions. However, after around 100–200 iterations, the rate of improvement significantly slows down for almost all algorithms, indicating that they are entering a fine-tuning phase of the search process or approaching local optima. At iteration 1000, Proposed-AFSA and AFSA appear to have the best function evaluations, with values ranging from 200 to 250.

This is a significant finding because it indicates these algorithms perform extremely well on this medium-sized instance. Modified AFSA 2 also performs well, converging slightly above the proposed algorithm and AFSA. Modified AFSA 3-method 3, modified AFSA 3-method 1, and modified AFSA1 all converge to higher values around 500-1000. Finally, modified AFSA 3-method 2 performs significantly worse than the other algorithms, becoming stuck at a function evaluation of around 1650. This indicates a significant lack of exploration or exploitation capability for this problem size.

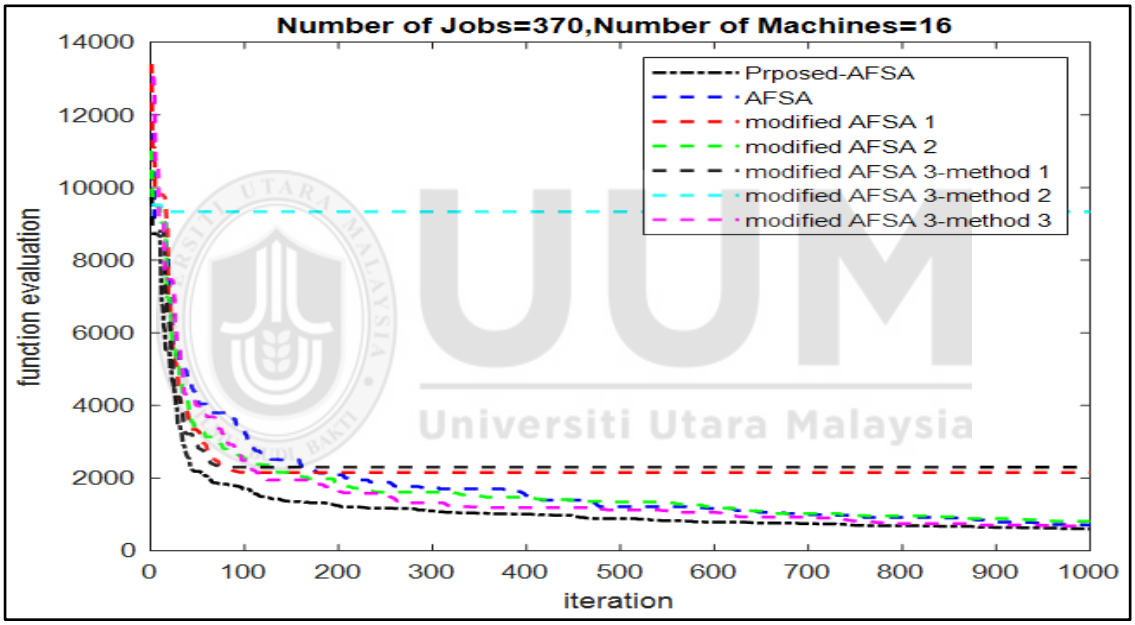


Figure 5.3. The performance of the proposed modified algorithm for sample large problem.

Figure 5.3 illustrates the graph for a sample large-sized problem with (N = 370, M = 16), it indicates that the problem is extremely complex and requires a much larger search space to find the best solutions. At iteration 0, all algorithms begin with very high function evaluations around 12000-13000, indicating the difficulty of finding good initial solutions in such a large search space. For most algorithms, function evaluation drops rapidly during the first 50–100 iterations. The modified AFSA 3-method 2 performs extremely poorly once

more. Modified AFSA 3-method 2 performs very badly once more, almost immediately getting trapped at a very high function evaluation (around 9500). This shows the futility of this algorithm. Other algorithms show increasingly gradual convergence, with some continue to improve slightly towards 1000 iterations, which indicates that more iterations may be beneficial. At iteration 1000, the proposed AFSA clearly emerges as the best performer, achieving the lowest function evaluation of approximately 500–600. Notably, the algorithm reaches this solution around iteration 750 and remains stable, showing no further improvement, until iteration 1000. Proposed-AFSA has one of the steepest initial drops, indicating a very fast initial search phase, as it achieves the best final result while maintaining a high convergence speed. This is a strong indication of its efficacy in large-scale problems. The performance of the AFSA and modified AFSA 2 are equivalent, converging to values around 800-900, which remains competitive but less effective to the proposed AFSA. The modified AFSA 3-method 3 converges to a value around 900-1000. Finally, modified AFSA 1 and modified AFSA 3-method 1 similarly converge to a significantly higher value, around 2200-2300.

Figures 5.2 and 5.3 illustrate a significant impact and confirm the efficiency of the proposed algorithm for medium and large-sized problem. This is evident from the smaller minimum values of the objective function, which expanded the convergence range and accelerated the algorithm towards the best feasible solutions. Additionally, the update process is guided to enhance the ability to search globally. As a result, the convergence results and quality of the solutions produced by the proposed algorithm improve as the number of jobs and iterations increases.

Finally, the Wilcoxon-rank sum test, a non-parametric test, is used to evaluate the performance of the modified algorithm. It is utilized to imply if there are major variations among the proposed algorithm and the compared algorithms. The results of the Wilcoxon signed-rank test for small, medium, and large instances, respectively, are summarized in Tables 5.5, 5.6, and 5.7. It compares the proposed modified AFSA with other modified AFSA versions. For all comparisons, the tables report the *p*-value, which represents the Wilcoxon test's significance level. The *p* -value serves as the basis for the decision rule, which is typically set at 0.05. The statistically significant difference in performance is indicated when the (*p* -value < 0.05), and if the (*p* -value > 0.05) the difference is statistically insignificant. The R+ and R- signify the sum of positive and negative ranks. Finally, Ind. represents indicator values (usually 1 or -1), which most likely indicate which method performed better. As a result, Ind +1 indicates that the proposed algorithm performs significantly better than the comparative algorithms, while Ind -1 indicates otherwise. The best results amongst the comparative algorithms for each problem are presented in grey.

Table 5.5

Results of the Wilcoxon signed rank test for small-sized problems

Problems		1	2	3	4	5	6	7	8	9	10
Proposed VS AFSA	***p*-value**	0.09770	0.01950	0.07810	0.18600	0.31600	0.13900	0.32200	0.49200	0.32200	0.29300
	R+	14.5	19	50.5	45	47	45	27	45	49	52
	R-	40.5	36	4.5	10	8	10	28	10	6	3
	Ind.	-1	-1	1	1	1	1	-1	1	1	1
Proposed VS Modified 1	***p*-value**	0.02148	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
	R+	52	55	55	55	55	55	55	55	55	55
	R-	3	0	0	0	0	0	0	0	0	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified 2	***p*-value**	0.57812	0.12500	0.10500	0.82800	0.98000	0.09960	0.48000	0.92200	0.37500	0.03710
	R+	33.5	14.5	47	40	40	52	45	45	45	52
	R-	21.5	40.5	8	15	15	3	10	10	10	3
	Ind.	1	-1	1	1	1	1	1	1	1	1
Proposed VS Modified 3 -method 1	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
	R+	55	55	55	55	55	55	55	55	55	55
	R-	0	0	0	0	0	0	0	0	0	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified 3 -method 2	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
	R+	55	55	55	55	55	55	55	55	55	55
	R-	0	0	0	0	0	0	0	0	0	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed VS Modified 3 -method 3	***p*-value**	0.37500	0.00390	0.00390	0.01367	0.00976	0.00195	0.00976	0.019531	0.00585	0.00976
	R+	47	54	54.5	52	54	55	52	52	54	52
	R-	8	1	0.5	3	1	0	3	3	1	3
	Ind.	1	1	1	1	1	1	1	1	1	1

Table 5.6

Results of the Wilcoxon signed rank test for medium-sized problems

Problems		1	2	3	4	5	6	7	8	9	10
Proposed	***p*-value**	1.00000	0.32000	0.27100	0.01560	0.02340	0.01170	0.11300	0.85900	0.94900	0.51600
VS	**R+**	32	50.5	49	18.5	14.5	14.5	30.5	41.5	42.5	26.5
AFSA	**R-**	23	4.5	6	36.5	40.5	40.5	24.5	13.5	12.5	28.5
	Ind.	1	1	1	-1	-1	-1	1	1	1	-1
Proposed	***p*-value**	1.00000	0.23400	0.28500	0.02540	0.02540	0.09770	0.41600	0.003910	0.01560	0.00195
VS	**R+**	36.5	45	50.5	52	54	50.5	40	54.5	54	55
Modified 1	**R-**	18.5	10	4.5	3	1	4.5	15	0.5	1	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed	***p*-value**	0.06250	0.44921	0.79300	0.27300	0.31100	0.06640	0.04300	0.19500	0.64500	0.09770
VS	**R+**	47.5	40	42.5	30.5	45	23	14.5	30.5	47	30.5
Modified 2	**R-**	7.5	15	12.5	24.5	10	32	40.5	24.5	8	24.5
	Ind.	1	1	1	1	1	-1	-1	1	1	1
Proposed	***p*-value**	0.50000	0.18750	0.19500	0.03520	0.00586	0.00586	0.06250	0.01170	0.00391	0.00195
VS	**R+**	37	48.5	48.5	53	54	54	49	54	54	55
Modified 3	**R-**	18	6.5	6.5	2	1	1	6	1	1	0
-method 1	**Ind.**	1	1	1	1	1	1	1	1	1	1
Proposed	***p*-value**	0.50000	0.12890	0.01562	0.01562	0.00195	0.00195	0.00390	0.00781	0.00195	0.00195
VS	**R+**	40.5	50.5	54	53	55	55	54.5	53	55	55
Modified 3	**R-**	14.5	4.5	1	2	0	0	0.5	2	0	0
-method 2	**Ind.**	1	1	1	1	1	1	1	1	1	1
Proposed	***p*-value**	1.00000	0.68750	0.92968	0.03906	0.06250	0.00195	0.02734	0.46875	0.42578	0.08007
VS	**R+**	36.5	41.5	42.5	23	19	0	19	37	42.5	54
Modified 3	**R-**	18.5	13.5	12.5	32	36	55	36	18	12.5	1
-method 3	**Ind.**	1	1	1	-1	-1	-1	-1	1	1	1

Table 5.7

Results of the Wilcoxon signed rank test for large-sized problems

Problems		1	2	3	4	5	6	7	8	9	10
Proposed	***p*-value**	0.19300	0.01370	0.13100	0.19300	0.62500	0.00195	0.77000	0.00586	0.16000	0.37500
VS	**R+**	45	54	49	49	45	55	34	54	52	45
AFSA	**R-**	10	1	6	6	10	0	21	1	3	10
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
VS	**R+**	55	55	55	55	55	55	55	55	55	55
Modified 1	**R-**	0	0	0	0	0	0	0	0	0	0
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed	***p*-value**	0.76953	0.00585	0.92200	0.49200	0.27500	0.00391	0.43200	0.19300	0.55700	0.69500
VS	**R+**	49	54	45	45	45	54	49	49	45	45
Modified 2	**R-**	6	1	10	10	10	1	6	6	10	10
	Ind.	1	1	1	1	1	1	1	1	1	1
Proposed	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
VS	**R+**	55	55	55	55	55	55	55	55	55	55
Modified 3	**R-**	0	0	0	0	0	0	0	0	0	0
-method 1	**Ind.**	1	1	1	1	1	1	1	1	1	1
Proposed	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
VS	**R+**	55	55	55	55	55	55	55	55	55	55
Modified 3	**R-**	0	0	0	0	0	0	0	0	0	0
-method 2	**Ind.**	1	1	1	1	1	1	1	1	1	1
Proposed	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.01367	0.00195	0.00195	0.00195
VS	**R+**	55	55	55	55	55	55	54	55	55	55
Modified 3	**R-**	0	0	0	0	0	0	1	0	0	0
-method 3	**Ind.**	1	1	1	1	1	1	1	1	1	1

Table 5.5 compares the results with other algorithms based on the Wilcoxon signed-ranks test for small-sized problems and shows that, *p*-values range from 0.0977 to 0.0293, with most values above 0.05 except for problem 2 and problem 10 (0.0195 and 0.0293, respectively), indicating that there is no significant difference for the majority of problems, except for problems 2 and 10. Furthermore, the R+ values are generally higher than R-, indicating that the proposed algorithm tends to outperform AFSA.

With the exception of problem 10 (0.0371), all of the problems in the proposed algorithm's comparison with Modified 2 had *p*-values above 0.05, indicating that there was no significant difference for these problems. Additionally, R+ values are higher than R-, indicating that the proposed algorithm is generally better.

The proposed algorithm compared Modified 1, Modified 3-method 1, Modified 3-method2, and Modified 3-method 3, found that *p*-values are extremely low and R+ values are higher than R- across all problems, indicating statistically significant differences in favor of the proposed algorithm.

Table 5.6 compares the results of other algorithms for medium-sized problems and shows that, when the proposed algorithm was compared to AFSA, the *p*-values ranged from 0.113 to 1.00 (all > 0.05), with the exception of problems 4, 5 and 6 having values (0.0156, 0.0234, 0.0171, respectively), and R+ > R- in most cases, indicating that the proposed algorithm was slightly better, but some problems showed significant differences while others do not. Furthermore, the proposed algorithm offers six problems with (*p*-values < 0.05 compared to Modified 1, and R+ > R- in all cases, indicating a better and significant difference for most problems. Meanwhile, the proposed algorithm has *p*-values that are mostly > 0.05 when compared to Modified 2, and R+ > R- in most cases, which indicates that it is better

with no significant difference. Modified 3-method 1, Modified 3-method 2, and Modified 3-method 3 had different *p*-values when compared by the proposed algorithm. Some *p*-values were greater than 0.05 and many were less than 0.05, and R+ was generally greater than R-, indicating a stronger and significant difference for most problems.

Finally, Table 5.7 presents the comparison results for large-sized problems, demonstrating that all comparisons had *p*-values significantly less than 0.05, indicating a statistically significant difference between the comparison algorithms and the proposed algorithm. Furthermore, the sum of R+ values is typically greater than R-, indicating that the proposed algorithm dominates comparison algorithms across all problems.

To sum up the discussions above, it is evident that the proposed algorithm's effectiveness stems from leveraging AFSA's advantages, including its multi-stage solution updates and search domain exploration. The algorithm enhances the exploitation and exploration phases by utilizing the best solution extracted from AFSA. Additionally, by adapting improved visual and step parameters, it balances global search capability with convergence speed and effectively applies AFSA to machine scheduling problems (MSPs). However, the proposed algorithm has limitations, particularly in achieving optimal results for machines 4 to 10. Thus, improvements are necessary for solving small-sized jobs and machine problems. While the stability of the proposed algorithm is commendable, further enhancements are required.

5.4 The Computational Results of the Proposed Hybrid Algorithm

This section discusses the performance of the proposed hybrid algorithm by comparing it with three sets of algorithms using the same performance evaluation criteria (Min, Max, Mean, STD, and Wilcoxon signed-rank test statistic) for small, medium, and large-sized problems. In particular, the first set of tests compared the computational results of the hybrid algorithm to standard AFSA and five modified AFSA algorithms. Meanwhile, the second set was conducted to compare the computational results of the hybrid algorithm to our proposed modified algorithm and five well-known single metaheuristic algorithms. Finally, the third set was conducted to compare the performance of the hybrid algorithm to three existing hybrid algorithms.

5.4.1 Computational Results with AFSA and Modified AFSA Algorithms

The first set of results compares the performance of the proposed hybrid algorithm to AFSA and five modified AFSA algorithms. Tables 5.8, 5.9, and 5.10 clarify the results of small, medium, and large-sized problems, respectively. The purpose of these comparisons is to evaluate the hybrid algorithm's efficiency using the same performance evaluation criteria, which is the final objective of this study.

Table 5.8

The computational results of the small-sized problems for the proposed hybrid algorithm with AFSA and five versions of modified AFSA

	N	M		Hybrid AFSA	AFSA	Modified 1	Modified 2	Modified 3 - method 1	Modified 3 - method 2	Modified 3 - method 3
1	**5**	**3**	**Mean**	13.50	13.80	13.60	13.75	13.75	13.75	13.75
			STD	0	0.34960	0.21081	0.42491	0.42491	0.35355	0.35355
			Min	13.50	13.50	13.50	13.50	13.50	13.50	13.50
			Max	13.50	14.50	14.00	14.50	14.50	14.50	14.50
2	**10**	**3**	**Mean**	26.50	28.50	29.15	28.50	28.50	28.60	29.05
			STD	0	1.35400	1.20300	1.08012	1.33333	1.19721	0.92646
			Min	26.50	26.50	28.00	27.00	27.00	26.50	27.00
			Max	26.50	30.50	31.00	30.50	30.50	30.50	30.00
3	**15**	**4**	**Mean**	29.60	32.95	33.80	32.65	33.75	34.20	32.60
			STD	0.21081	1.32182	0.85634	1.15590	1.49536	1.71917	0.69920
			Min	29.50	30.50	32.50	31.50	31.50	31.00	31.50
			Max	30.00	35.00	35.00	35.00	36.00	36.50	34.00
4	**20**	**4**	**Mean**	42.70	46.60	49.35	47.55	51.10	51.75	47.45
			STD	0.82327	1.64654	2.69825	1.80200	2.01108	2.30036	1.89223
			Min	42.00	44.00	46.00	44.50	48.50	48.50	45.50
			Max	45.00	49.50	54.50	50.00	54.50	55.00	52.00
5	**25**	**5**	**Mean**	38.95	43.90	47.30	43.65	47.35	48.55	42.75
			STD	1.67414	1.72884	2.22610	1.313392	1.43468	3.38665	1.51382
			Min	37.00	41.00	45.00	41.50	45.50	42.00	40.00
			Max	41.00	47.00	52.50	45.00	50.00	53.00	44.50
6	**30**	**5**	**Mean**	46.80	53.20	56.35	53.25	58.80	58.35	52.35
			STD	1.60208	2.38280	2.88723	3.06639	4.80277	3.41605	1.91557
			Min	45.00	50.50	52.50	49.50	53.00	55.00	49.50
			Max	48.50	57.00	60.00	57.50	68.00	65.50	55.50
7	**35**	**6**	**Mean**	44.55	49.25	53.35	50.05	54.40	53.85	49.45
			STD	2.46587	1.29636	2.17370	1.25720	1.77638	2.27364	1.21220
			Min	42.00	48.00	50.50	48.50	51.50	51.50	47.50

			Max	48.00	51.50	58.00	52.50	58.50	58.50	52.00
8	**40**	**6**	**Mean**	53.00	56.90	61.70	57.10	60.95	63.60	57.25
			STD	4.71404	1.37032	1.94650	0.93689	1.44241	2.72641	1.39940
			Min	48.50	54.00	59.00	55.50	59.00	59.00	55.00
			Max	61.00	59.00	64.50	58.50	63.50	68.50	60.00
9	**45**	**7**	**Mean**	51.10	54.00	59.05	54.80	60.20	62.25	54.50
			STD	4.50801	1.58113	1.57144	1.13529	2.84995	3.45004	1.35400
			Min	46.50	52.00	57.00	53.00	57.50	57.50	52.50
			Max	61.00	57.00	62.50	57.00	66.50	68.00	56.00
10	**50**	**7**	**Mean**	55.10	62.45	65.25	61.70	67.90	71.85	62.80
			STD	3.34829	0.64334	1.78341	1.13529	4.04694	4.48485	0.91893
			Min	52.00	61.50	62.00	59.50	62.50	66.00	61.50
			Max	61.50	63.50	68.00	63.50	74.50	82.50	64.50

Table 5.9

The computational results of the medium-sized problems for the proposed hybrid algorithm with AFSA and five versions of modified AFSA

	N	**M**		**Hybrid AFSA**	**AFSA**	**Modified 1**	**Modified 2**	**Modified 3 -method 1**	**Modified 3 - method 2**	**Modified 3 - method 3**
1	**60**	**8**	**Mean**	67.10	72.95	83.15	71.85	85.85	89.85	74.55
			STD	6.74866	1.93577	6.91636	1.71674	7.48350	12.53229	1.03949
			Min	61.00	70.50	74.00	70.00	73.50	73.00	73.00
			Max	79.50	76.50	98.00	74.50	97.00	116.50	75.50
2	**80**	**8**	**Mean**	78.60	88.65	100.20	88.95	100.30	110.70	92.45
			STD	3.15171	1.73285	6.95301	1.30064	5.65783	12.60555	2.81316
			Min	74.50	85.50	91.50	86.50	94.00	93.00	88.50
			Max	82.00	91.50	117.50	90.50	114.00	127.50	96.00
3	**100**	**9**	**Mean**	95.70	111.65	140.00	112.85	147.45	162.00	121.60
			STD	6.05621	6.09667	18.79125	3.92322	10.67304	26.94438	7.08989
			Min	86.50	105.50	119.50	107.50	131.50	134.00	113.00

			Max	105.50	122.00	185.00	121.00	161.50	224.00	133.50
4	**120**	**9**	**Mean**	119.80	139.05	169.45	142.10	178.95	202.40	150.05
			STD	7.72873	6.48266	18.24136	10.81357	13.66758	48.97947	9.50570
			Min	109.50	132.50	144.50	129.50	160.50	139.00	138.00
			Max	136.00	152.50	196.00	164.50	206.00	270.00	166.50
5	**140**	**10**	**Mean**	133.50	156.90	219.50	152.40	221.85	292.05	182.70
			STD	7.96868	12.66840	30.21405	6.50128	26.85873	57.04308	15.39877
			Min	122.00	145.00	170.50	144.00	184.00	206.00	149.50
			Max	147.00	186.00	272.50	164.00	276.00	389.50	196.50
6	**160**	**10**	**Mean**	166.20	176.20	322.90	178.80	372.85	393.50	217.20
			STD	15.90108	8.75975	47.27449	7.88528	91.44337	83.21992	18.46799
			Min	143.50	159.50	228.00	171.00	255.00	283.00	188.50
			Max	198.00	187.50	384.50	197.50	523.00	553.50	252.50
7	**180**	**11**	**Mean**	144.05	186.85	290.10	175.40	288.80	381.00	201.00
			STD	11.46601	23.74750	50.27579	11.17735	40.48470	75.54101	14.46067
			Min	130.00	157.00	234.50	162.50	242.50	285.50	181.50
			Max	164.50	231.50	384.00	202.00	363.00	537.50	223.00
8	**200**	**11**	**Mean**	180.90	213.40	461.20	212.75	519.00	780.25	265.40
			STD	21.41105	21.37210	66.47397	20.95796	117.33593	232.53342	46.04032
			Min	152.50	180.50	363.50	189.50	379.00	464.50	213.00
			Max	227.50	250.50	539.00	249.00	754.50	1050.00	361.50
9	**220**	**12**	**Mean**	173.30	222.15	427.80	235.20	599.55	796.80	303.65
			STD	20.13040	18.03390	72.04674	30.75458	121.73250	260.27455	51.68389
			Min	152.50	196.00	325.00	189.50	412.00	461.50	233.00
			Max	218.50	248.00	547.00	283.50	803.50	1350.50	405.50
10	**240**	**12**	**Mean**	221.75	279.20	764.70	287.15	818.30	1314.50	399.50
			STD	15.60849	39.44130	221.98138	35.32472	121.37206	332.50329	50.21066
			Min	190.50	228.00	502.00	254.50	596.50	832.00	322.00
			Max	239.50	370.00	1226.50	371.00	969.00	1950.50	465.00

Table 5.10

The computational results of the large-sized problems for the proposed hybrid algorithm with AFSA and five versions of modified AFSA

	N	M		Hybrid AFSA	AFSA	Modified 1	Modified 2	Modified 3 -method 1	Modified 3 -method 2	Modified 3 - method 3
1	**250**	**13**	**Mean**	204.60	265.00	565.05	272.65	620.50	1267.90	311.75
			STD	20.55993	28.33039	117.25078	36.66064	115.88380	433.85537	46.51299
			Min	174.50	220.50	298.50	239.00	481.50	7250.00	234.50
			Max	234.00	304.50	722.00	363.50	799.50	2115.50	374.50
2	**270**	**13**	**Mean**	237.90	466.15	1513.40	476.25	1670.35	2785.35	665.80
			STD	27.86355	46.45670	138.36642	52.91043	334.91317	939.46882	83.96897
			Min	193.50	403.00	1293.00	371.00	1247.50	1734.00	580.50
			Max	290.00	558.00	1717.00	560.50	2167.50	4526.50	818.50
3	**290**	**14**	**Mean**	252.30	469.20	1211.95	445.60	1246.65	3027.80	595.75
			STD	28.07945	48.05852	279.78339	91.79621	177.38032	441.91818	91.46773
			Min	220.50	407.50	900.00	336.50	890.00	2252.50	419.00
			Max	295.00	565.00	1664.50	639.50	1456.00	3584.50	700.00
4	**310**	**14**	**Mean**	267.40	536.10	1704.05	580.65	1889.65	5609.75	906.50
			STD	31.30122	67.81092	199.73433	62.89808	391.18637	974.65886	169.80773
			Min	230.00	450.00	1436.50	486.50	1410.50	4292.00	688.00
			Max	326.00	652.50	2002.00	677.50	2705.00	7902.50	1208.50
5	**330**	**15**	**Mean**	246.95	657.50	1816.35	657.90	2229.75	7866.60	851.75
			STD	17.06271	136.85616	191.99566	170.93530	347.14592	610.64914	146.35179
			Min	219.00	480.50	1429.00	417.00	1667.50	6816.00	654.50
			Max	267.50	875.00	2052.50	1059.00	2628.50	8533.50	1120.00
6	**350**	**15**	**Mean**	318.25	1228.80	3252.65	1286.70	3370.05	10482.10	1463.65
			STD	23.04132	176.01171	389.50125	201.81925	303.00545	774.78440	178.72543
			Min	294.00	932.00	2638.00	928.50	2936.00	8894.50	1209.50

			Max	374.50	1564.00	3892.00	1610.50	3829.50	11775.50	1798.00
7	**370**	**16**	**Mean**	287.05	653.70	1919.55	613.95	1960.45	7680.45	737.30
			STD	37.14644	164.22870	362.88638	59.39297	462.07938	715.92855	115.63789
			Min	230.00	444.00	1209.50	526.00	1476.50	6010.00	565.00
			Max	349.50	895.00	2608.50	687.00	2767.00	8327.00	940.50
8	**390**	**16**	**Mean**	294.35	809.95	2126.00	695.85	2414.75	9171.30	1126.75
			STD	23.80131	239.53687	361.67396	123.75290	319.82349	469.43815	330.03444
			Min	259.50	611.00	1631.00	521.50	1988.00	8273.00	646.50
			Max	328.50	1360.00	2648.00	944.50	2947.50	9967.00	1847.00
9	**410**	**17**	**Mean**	309.50	934.60	2849.30	1047.60	3060.00	10775.40	1115.00
			STD	28.87136	150.78918	501.88850	210.84259	362.782839	1087.987791	160.66683
			Min	267.50	687.50	2225.50	810.00	2515.50	8925.50	920.50
			Max	364.00	1216.00	3654.50	1420.50	3722.00	12508.50	1398.00
10	**430**	**17**	**Mean**	303.20	1322.60	3845.70	1405.10	4136.55	13283.45	1836.35
			STD	23.25606	178.15283	517.64156	155.27981	547.38577	892.43115	388.62428
			Min	270.00	1017.00	3117.50	1188.00	3501.50	12025.50	1189.50
			Max	345.50	1629.50	4865.00	1644.50	5511.00	14910.50	2399.50

The comparison of the proposed hybrid algorithm with AFSA and other modified algorithms indicates that the proposed hybrid algorithm outperforms all the comparative algorithms evaluated. It provides excellent solutions for all problems of all sizes. Figures 5.4, 5.5, and 5.6 illustrate the efficacy of the proposed hybrid algorithm across all test problem sizes within the provided sample graph.

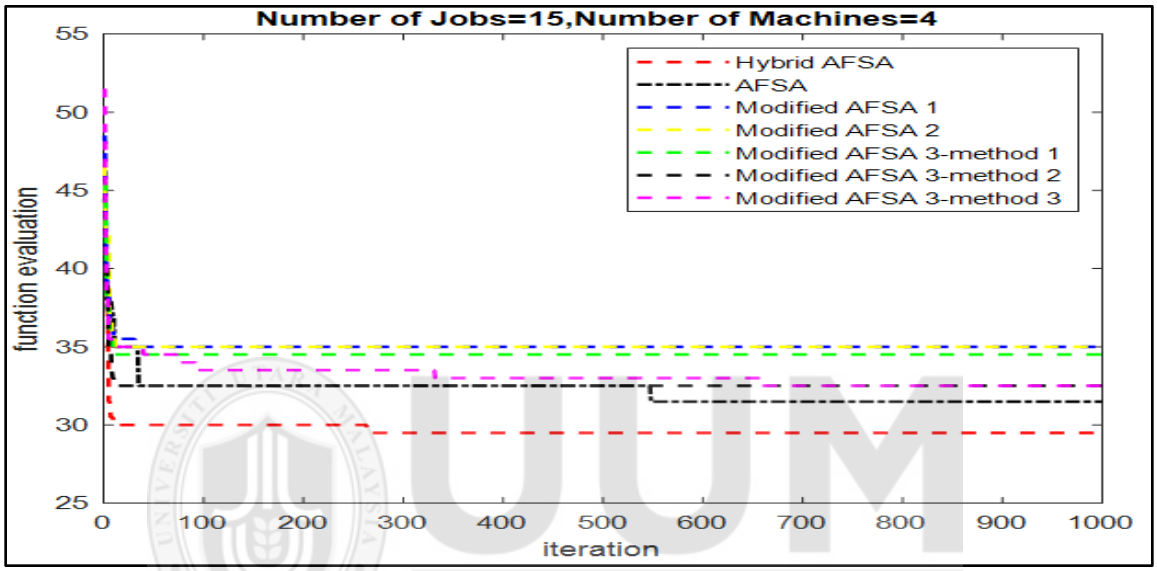


Figure 5.4. The performance of hybrid AFSA with standard AFSA and other modified AFSA algorithms for small-sized sample problem.

Figure 5.4 shows the graph for a small-size problem instance (N=15, M=4), which displays a comparison of the efficiency and effectiveness of performance results of different AFSA modifications, including a proposed hybrid AFSA algorithm.

At iteration 0, all algorithms start with high function evaluations, most above 45, illustrating that their initial solutions have not yet been optimized, and also most algorithms show very rapid initial convergence, with significant drops in function evaluation occurring within the first 50-100 iterations.

During 20-30 iterations, the hybrid AFSA (red dashed) reaches a near-final solution quality almost instantly, with exceptional convergence. Additionally, modified AFSA 3-method 2 (black line) and AFSA (black dashed) converge rapidly. At iteration 1000, hybrid AFSA clearly achieves the best function evaluation, settling at 29-30. This means it finds the best solution among all tested algorithms for this particular problem size. AFSA and Modified AFSA 3-method 2 converge on the next best values, which are around 33-31.

The modified AFSA 3-method 3 (magenta dashed) converges at around 31-32. While Modified AFSA 2 (yellow dashed) and Modified AFSA 3-method 1 (green dashed) converge to higher values of around 32-33. Modified AFSA 1 (blue dashed) performs the worst, converging to the highest value of around 34.

While many algorithms have similar initial convergence speed, the hybrid AFSA is the only one that successfully converges to the best solution at around 300 iteration and demonstrates the fastest and most effective convergence. Most algorithms achieve stable convergence after a rapid descent, which flattens their curves. When the hybrid AFSA reaches its best solution, it becomes exceptionally stable. Meanwhile, AFSA improves slightly around iteration 550, indicating that it may have escaped a local optimum or discovered a better solution through additional exploration.

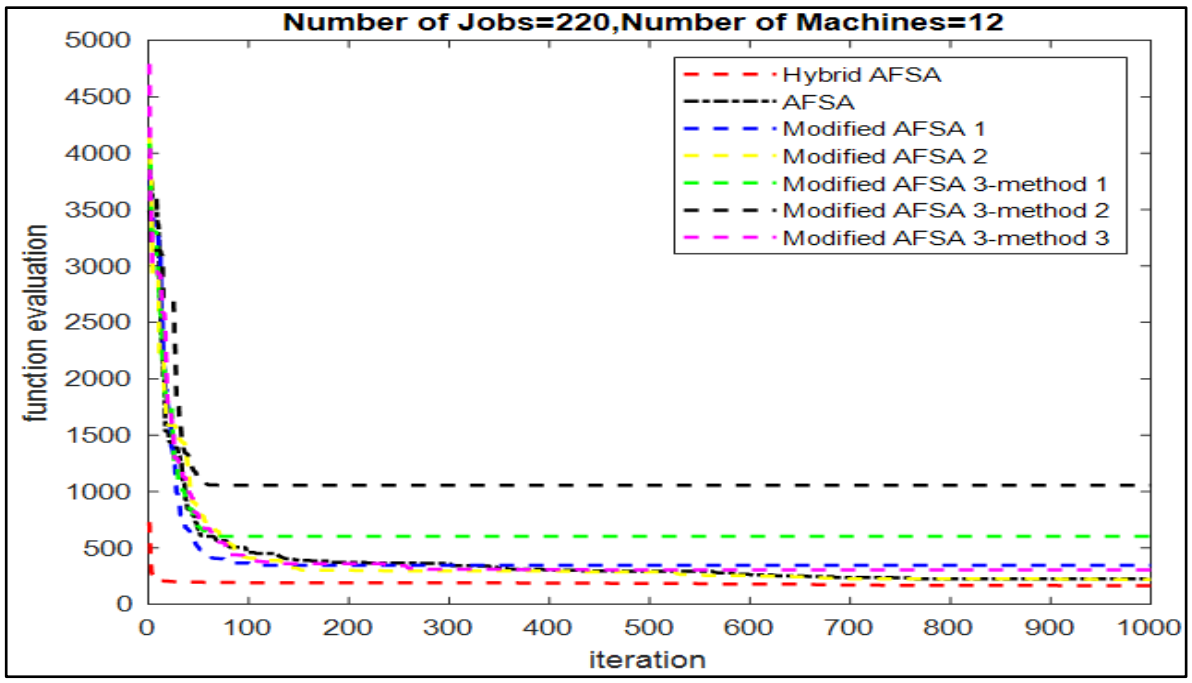


Figure 5.5. The performance of hybrid AFSA with standard AFSA and other modified AFSA algorithms for medium-sized sample problem.

The graph for a medium-sized problem instance (N=220, M=12) is shown in Figure 5.5. All algorithms start high function evaluations at iteration 0 (around 4500–5000), indicating that the initial solutions are far from the best. In the first 50-100 iterations, all algorithms show a sharp decline in function evaluation, indicating their ability to rapidly improve upon initial solutions. The hybrid AFSA exhibits exceptionally quick convergence, dropping to a very low function evaluation almost immediately and then maintaining that low value until the end of iterations. Thus, at iteration 1000, hybrid AFSA is clearly the best performer, with the lowest function rating around 150-200. It maintains excellent stability by quickly finding and maintaining a high-quality solution.

AFSA, Modified AFSA 1, Modified AFSA 2, and Modified AFSA 3-method 3 all converge to similar values, ranging from approximately 200 to 250, and demonstrate relatively stable convergence after the initial phase.

Modified AFSA 3-method 1 converges to a higher value, around 600, while Modified AFSA 3-method 2 performs the worst, converging to a value around 1050, both exhibit poor stability and appear to be stuck in local optima at much higher function evaluations.

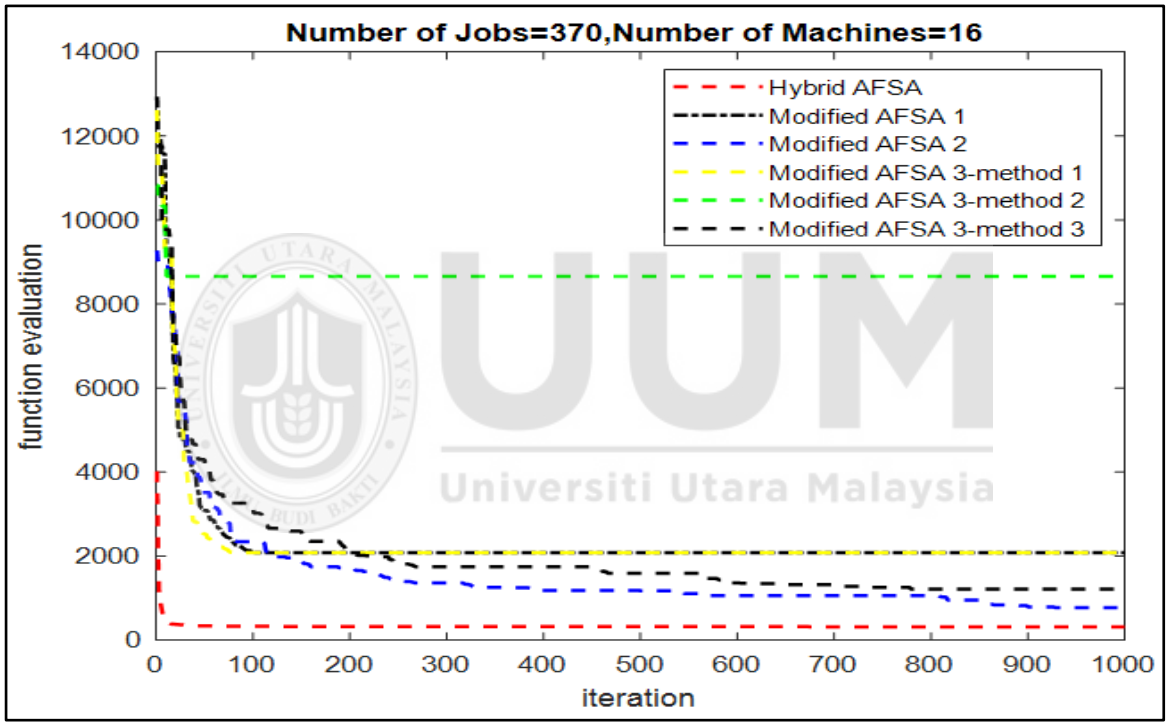


Figure 5.6. The performance of hybrid AFSA with standard AFSA and other modified AFSA algorithms for large sample problem.

The graph for a large-sized problem sample (N=370, M=16) is shown in Figure 5.6. It evaluates the ability of the hybrid algorithm to efficiently explore large search spaces, avoid local optimal solutions, and find high-quality solutions to a large-size problem.

At iteration 0, all algorithms begin with high evaluations (between 10000-13000), and the initial solutions are far from the best. Most algorithms show a rapid drop in function evaluation within the first 50-100 iterations, indicating that they quickly find better solutions. However, the rate of improvement then slows significantly. The hybrid AFSA shows remarkably rapid and aggressive convergence. The function evaluation falls to a very low level almost instantly (in the first 20–30 iterations) and then remains stable. The modified AFSA 3-method 2 performs very badly until the iterations end, becoming stuck at a very high function evaluation (around 8600) almost immediately. This indicates its total lack of effectiveness for larger problems.

At iteration 1000, hybrid AFSA performs best overall, converging to the lowest function evaluation, which is around 200-250. Modified AFSA 2 converges to the next best value, which is around 600–700. Modified AFSA 3-method 3 converges to approximately 800-900. Finally, Modified AFSA 1 and Modified AFSA 3-method 1 converge to approximately 2100-2200.

In summary, the proposed hybrid algorithm performs better as the number of jobs and iterations increases for all test problem sizes. Additionally, it outperforms other modified algorithms in terms of convergence to the best solution.

Tables 5.11, 5.12, and 5.13 summarize the results of the Wilcoxon signed-rank test for small, medium, and large instances, respectively, of the proposed hybrid algorithm with other modified algorithms.

Table 5.11

Results of the Wilcoxon signed rank test for the proposed hybrid algorithm with other modified algorithms for small-sized problems

Problems		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Hybrid	***p*-value**	0.06250	0.00781	0.00195	0.00391	0.00195	0.00195	0.00391	0.03710	0.09770	0.00195
AFSA	**R+**	47.5	53.5	55	54.5	55	55	54.5	52	50.5	55
VS	**R-**	7.5	1.5	0	0.5	0	0	0.5	3	4.5	0
AFSA	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.25000	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00391	0.00195
AFSA	**R+**	41	55	55	55	55	55	55	55	54.5	55
VS	**R-**	14	0	0	0	0	0	0	0	0.5	0
Modified 1	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.12500	0.00195	0.00195	0.00195	0.01170	0.00391	0.00781	0.00195	0.62500	0.00391
AFSA	**R+**	44.5	55	55	55	53	54	53	55	45	54
VS	**R-**	10.5	0	0	0	2	1	2	0	10	1
Modified 2	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.25000	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	41	55	55	55	55	55	55	55	55	55
VS	**R-**	14	0	0	0	0	0	0	0	0	0
Modified 3 method 1	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid AFSA	***p*-value**	0.12500	0.00390	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
VS	**R+**	44.5	54.5	55	55	55	55	55	55	55	55
Modified 3	**R-**	10.5	0.5	0	0	0	0	0	0	0	0
-method 2	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid AFSA	***p*-value**	0.50000	0.01562	0.00195	0.00195	0.14100	0.00391	0.00391	0.00195	0.18800	0.00391
VS	**R+**	37	51.5	55	55	47	54	54	55	49	54.5
Modified 3	**R-**	18	3.5	0	0	8	1	1	0	6	0.5
-method 3	**Ind.**	1	1	1	1	1	1	1	1	1	1

Table 5.12

Results of the Wilcoxon signed rank test for the proposed hybrid algorithm with other modified algorithms for medium-sized problems

Problems		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Hybrid	***p*-value**	0.07031	0.00195	0.00195	0.00391	0.00195	0.12500	0.00195	0.00586	0.00977	0.00195
AFSA	**R+**	52	55	55	54	55	52	55	54	52	55
VS	**R-**	3	0	0	1	0	3	0	1	3	0
AFSA	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.00195	0.00390	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	54	55	55	55	55	55	55	55	55
VS	**R-**	0	1	0	0	0	0	0	0	0	0
Modified 1	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00977	0.08400	0.00195	0.00391	0.00195	0.00586
AFSA	**R+**	55	54	55	55	54	49	55	54	55	54
VS	**R-**	0	1	0	0	1	6	0	1	0	1
Modified 2	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	55	55	55	55	55	55	55	55
VS	**R-**	0	0	0	0	0	0	0	0	0	0
Modified 3 method 1	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid AFSA	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
VS	**R+**	55	55	55	55	55	55	55	55	55	55
Modified 3	**R-**	0	0	0	0	0	0	0	0	0	0
-method 2	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid AFSA	***p*-value**	0.00195	0.00390	0.00391	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
VS	**R+**	55	54	54	55	55	55	55	55	55	55
Modified 3	**R-**	0	1	1	0	0	0	0	0	0	0
-method 3	**Ind.**	1	1	1	1	1	1	1	1	1	1

Table 5.13

Results of the Wilcoxon signed rank test for the proposed hybrid algorithm with other modified algorithms for large-sized problems

Problems		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	55	55	55	55	55	55	55	55
VS	**R-**	0	0	0	0	0	0	0	0	0	0
AFSA	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	55	55	55	55	55	55	55	55
VS	**R-**	0	0	0	0	0	0	0	0	0	0
Modified 1	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	55	55	55	55	55	55	55	55
VS	**R-**	0	0	0	0	0	0	0	0	0	0
Modified 2	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	55	55	55	55	55	55	55	55
VS	**R-**	0	0	0	0	0	0	0	0	0	0
Modified 3 method 1	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid AFSA	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
VS	**R+**	55	55	55	55	55	55	55	55	55	55
Modified 3	**R-**	0	0	0	0	0	0	0	0	0	0
-method 2	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid AFSA	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
VS	**R+**	55	55	55	55	55	55	55	55	55	55
Modified 3	**R-**	0	0	0	0	0	0	0	0	0	0
-method 3	**Ind.**	1	1	1	1	1	1	1	1	1	1

For small, medium, and large-sized problems and across ten problem instances, the proposed hybrid algorithm was compared with AFSA and other modified AFSA algorithms in the Wilcoxon signed-ranks test comparison tables. The proposed hybrid algorithm consistently demonstrates statistically significant differences (p-values < 0.05). Furthermore, it consistently achieved higher R+ values, indicating superior performance over AFSA and its modified versions.

5.4.2 Computational Results of Single Metaheuristics Algorithms

The second set of results compares the performance of the proposed hybrid algorithm with the proposed modified algorithm and five single metaheuristic algorithms using the same performance evaluation criteria.

The five metaheuristic algorithms involved in the comparison are the Firefly Algorithm (FA), Tabu Search (TS), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA) algorithms.

Notably, fine-tuning parameters is a set of experiments to determine the effect of parameters on the hybrid algorithm's performance. These experiments aim to determine the best possible values for all parameters utilized in the algorithms before their comparison. The experiments were conducted iteratively, and all parameters for each experimental set were examined. Notably, the proposed algorithm can work better if the control parameters for each algorithm are fine-tuned. However, determining the correct parameter settings for each algorithm can be lengthy. All comparative algorithms were employed for the test problem, with the maximum number of iterations set at 1,000 and the population size set at 40 for each algorithm to ensure a fair comparison. Note that each algorithm has been run ten times

on each test for all instances of varying problem sizes. Table 5.14 summarizes the recommended settings for all proposed comparative algorithms applied in this research.

Table 5.14

Parameter settings for all algorithms in the comparison

Algorithm	Parameters
Proposed hybrid AFSA	nr = 10, σ = 0.6, μ = 0.3, step1 = stepmin1 = 1, step2 = stepmin2 = 2, visual = visual min = 30, Try_number = 10, It (iteration number of neighbors) = 10.
FA	gamma (light coefficient) = 1, beta (attraction coefficient) = 2, alpha (mutation coefficient) = 0.2, alpha_damp (mutation coefficient damping ratio) = 0.98
GA	pc (crossover rate) = 0.4, pm (mutation rate) = 0.8, beta (selection pressure of roulette wheel) = 5
PSO	w (inertia weight) = 1, wdamp (inertia weight damping ratio) = 0.99, c1 (personal learning coefficient) = 1.5, c2 (global learning coefficient) = 2.0
SA	T_0 (initial temperature) = 10, alpha (temperature damping rate) = 0.97
TS	TL = nSwap+nReversion+nInsertion

Table 5.15 presents the hybrid AFSA test results for small-sized problems. The proposed hybrid algorithm produces the best minimum values. Note that it outperforms the proposed modified and all other compared algorithms for all instances.

Table 5.15

The computational results of hybrid AFSA with the proposed modified algorithm and five single metaheuristics algorithms for small-sized problems

	N	M		Hybrid AFSA	Proposed AFSA	FA	GA	PSO	SA	TS
1	**5**	**3**	**Mean**	14.15	14.95	14.00	14.15	14.15	14.75	14.60
			STD	0.47434	0.83166	0	0.47434	0.47434	0.79056	0.77459
			Min	14.00	14.00	14.00	14.00	14.00	14.00	14.00
			Max	15.50	16.00	14.00	15.50	15.50	15.50	15.50
2	**10**	**3**	**Mean**	28.60	29.40	28.50	28.50	28.60	29.60	28.50
			STD	0.31622	0.77459	0	0	0.31622	0.96609	0
			Min	28.50	28.50	28.50	28.50	28.50	28.50	28.50
			Max	29.50	30.50	28.50	28.50	29.50	30.50	28.50
3	**15**	**4**	**Mean**	28.60	31.85	29.45	30.60	29.05	29.35	30.10
			STD	0.21081	0.91439111	0.28382310	2.45854518	0.28382310	1.66749979	1.55991452
			Min	28.50	30.50	29.00	28.50	28.50	28.50	28.50
			Max	29.00	33.50	30.00	36.00	29.50	34.00	33.00
4	**20**	**4**	**Mean**	39.85	44.60	40.40	45.70	41.25	43.80	43.30
			STD	1.15590	1.86785	0.51639	2.91738	1.60294	3.35989	2.51881
			Min	39.00	41.00	39.50	41.00	40.00	39.00	40.00
			Max	42.00	47.50	41.00	52.50	45.00	50.00	49.50
5	**25**	**5**	**Mean**	39.40	44.10	40.30	45.75	41.80	41.95	42.20
			STD	1.22020	1.28668	0.82327	4.51078	1.96072	3.66249	2.48551
			Min	38.00	42.50	39.50	40.50	40.00	38.00	38.50
			Max	40.50	46.50	42.00	53.50	46.50	48.50	46.00
6	**30**	**5**	**Mean**	48.20	53.35	47.60	55.40	50.70	49.85	49.90
			STD	2.03032	2.06895	0.56764	4.86369	1.75119	4.73198	2.19595
			Min	45.50	51.00	46.50	49.00	48.50	46.00	47.00
			Max	50.50	58.00	48.50	63.50	54.00	62.50	54.50
7	**35**	**6**	**Mean**	48.50	54.50	48.15	58.80	52.75	52.60	49.05
			STD	2.56038	1.45296	0.85146	3.58391	3.61516	6.62822	1.97835

			Min	44.50	51.00	47.00	51.50	49.00	45.50	47.00
			Max	50.50	56.00	49.500	65.500	59.500	66.00	52.50
8	**40**	**6**	**Mean**	55.15	61.65	56.25	65.95	62.05	56.20	58.30
			STD	2.93494	1.85666	1.03413	4.95787	3.72267	2.58413	3.86005
			Min	52.50	59.00	54.50	59.00	56.00	52.50	54.00
			Max	59.50	64.50	57.00	74.00	68.50	58.50	64.00
9	**45**	**7**	**Mean**	51.45	56.90	51.00	63.70	57.70	52.15	52.95
			STD	3.26981	1.24275	1.69967	3.98748	4.95647	2.73911	3.80387
			Min	47.50	55.00	49.00	55.00	52.00	47.50	50.00
			Max	55.00	59.00	55.00	69.00	66.50	54.50	61.50
10	**50**	**7**	**Mean**	57.70	66.05	58.20	73.85	69.40	60.55	62.45
			STD	3.44963	2.44324	1.20646	7.13383	4.22821	5.41320	4.70490
			Min	54.00	63.50	56.00	65.00	64.00	55.00	54.50
			Max	63.50	71.50	60.50	89.00	78.50	71.00	69.00

Table 5.16 illustrates the hybrid algorithm test results for medium-sized problems. The hybrid algorithm outperforms the proposed modified algorithm for all instances and produces the best minimum values. Compared to other algorithms, it produces the best minimum values for seven instances: 1, 2, 3, 4, 6, 7, and 10, respectively. In particular, SA obtains the best value of one instance when (N = 140, M = 10), while FA obtains the best value of two instances when (N = 200, M = 11), and (N = 220, M = 12).

Table 5.16

The computational results of the hybrid AFSA with the proposed modified algorithm and five single metaheuristics algorithms for medium-sized problems

	N	M		Hybrid AFSA	Proposed AFSA	FA	GA	PSO	SA	TS
	60	**8**	**Mean**	63.30	74.35	63.05	81.90	76.95	68.60	69.75
1			**STD**	3.94546	1.95859	1.44241	8.62747	9.68805	6.25299	5.95469
			Min	57.00	72.00	61.50	73.00	68.00	58.00	59.50
			Max	70.00	77.50	65.50	99.50	96.00	77.50	78.00
	80	**8**	**Mean**	78.05	89.40	85.80	104.70	102.20	86.00	95.80
2			**STD**	4.86169	1.42984	6.20573	8.11445	7.50999	7.90920	7.07970
			Min	73.50	86.00	77.50	91.50	87.50	76.50	86.50
			Max	89.50	91.00	100.50	115.00	111.50	96.00	112.50
	100	**9**	**Mean**	93.35	106.80	97.65	120.30	116.40	94.20	108.80
3			**STD**	4.77289	3.11091	3.42417	10.88373	5.78695	4.60796	5.69697
			Min	88.00	104.00	94.00	111.00	107.00	90.50	102.00
			Max	104.00	114.50	102.50	138.00	125.00	104.50	117.00
	120	**9**	**Mean**	116.55	132.75	114.95	157.35	147.45	122.70	146.30
4			**STD**	7.35772	3.83152	2.87179	15.0703	8.62312	14.57394	17.83130
			Min	107.00	129.00	113.00	138.50	138.00	107.00	121.50
			Max	130.00	141.00	122.50	180.50	162.00	146.00	172.00
	140	**10**	**Mean**	121.20	141.45	125.75	163.70	157.60	126.70	159.15
5			**STD**	4.80277	6.28689	3.70622	17.96323	14.77385	15.17161	17.15622
			Min	114.50	132.50	117.50	146.00	140.00	111.50	134.50
			Max	129.00	151.00	129.50	211.00	177.50	156.00	184.50
	160	**10**	**Mean**	160.95	173.30	145.40	190.90	201.00	175.50	174.95
6			**STD**	17.91097	15.98645	4.73051	10.66615	14.20876	21.56385	13.16234
			Min	137.00	160.00	137.50	175.00	176.00	143.50	153.50
			Max	185.00	206.50	151.50	208.50	225.50	210.00	192.00
	180	**11**	**Mean**	154.75	183.30	145.35	207.05	194.80	163.35	179.25
7			**STD**	20.845	12.13168	4.35284	14.10959	16.85427	18.63546	7.146288

			Min	128.00	159.50	139.50	188.00	176.00	134.00	168.00
			Max	181.50	197.50	152.50	229.50	228.50	195.50	195.00
	200	**11**	**Mean**	197.35	232.10	170.00	232.65	237.15	218.10	208.65
8			**STD**	28.47323	22.90657	4.05517	16.72664	13.98818	23.37115	15.42193
			Min	166.50	198.00	164.00	211.00	2180	190.00	185.50
			Max	252.50	265.00	178.50	266.50	255.50	261.00	230.50
	220	**12**	**Mean**	189.45	262.30	166.25	245.40	236.95	204.50	202.60
9			**STD**	19.26922	33.59828	5.21882	19.55448	14.30704	21.17125	12.98032
			Min	166.00	210.00	159.00	218.00	211.50	167.50	184.00
			Max	217.50	316.50	174.50	281.50	252.50	233.50	219.50
10	**240**	**12**	**Mean**	194.30	235.55	184.90	250.80	257.85	232.50	208.80
			STD	27.60555	26.88912	5.99444	21.10319	27.54798	32.10745	10.75794
			Min	164.00	213.00	172.00	223.50	218.00	189.50	192.00
			Max	245.00	308.00	191.50	291.00	305.50	298.00	235.00

Table 5.17

The computational results of the hybrid AFSA with the proposed modified algorithm and five single metaheuristics algorithms for large-sized problems

	N	**M**		**Hybrid AFSA**	**Proposed AFSA**	**FA**	**GA**	**PSO**	**SA**	**TS**
	250	**13**	**Mean**	224.15	337.60	188.30	286.70	297.40	232.85	217.65
1			**STD**	33.26163	61.85502	7.13831	17.92608	27.48514	23.37621	10.46170
			Min	184.00	263.00	179.00	269.00	263.50	189.50	202.50
			Max	280.50	475.00	198.50	317.00	343.50	283.50	232.50
	270	**13**	**Mean**	231.95	347.15	196.40	291.85	347.55	263.20	233.30
2			**STD**	36.12744	52.96175	5.60654	28.15735	26.142186	38.78301	12.90391
			Min	180	260.00	186.50	240.00	308.00	211.50	220.50
			Max	302.50	463.50	203.50	334.50	391.50	334.00	253.50
	290	**14**	**Mean**	240.25	360.95	209.80	318.00	348.45	264.80	232.25
3			**STD**	20.4318	48.79005	8.26034	27.555	62.33888	33.22081	8.63536
			Min	209.50	295.50	200.50	281.50	257.50	210.50	219.50

			Max	264.50	439.00	224.50	372.50	437.50	332.50	244.50
	310	**14**	**Mean**	257.45	479.95	213.95	303.30	336.35	270.45	241.40
4			**STD**	31.48849	99.71248	7.01367	21.08738	33.93052	24.12864	9.35948
			Min	209.50	323.50	205.00	274.50	288.00	228.00	230.50
			Max	304.50	606.50	225.50	334.50	400.00	313.00	257.50
	330	**15**	**Mean**	277.20	613.00	227.85	357.10	488.55	282.45	256.80
5			**STD**	32.00972	141.00374	8.60248	33.27395	42.81903	29.55733	13.27947
			Min	244.50	370.00	218.00	319.00	413.00	250.00	238.00
			Max	343.00	899.50	240.50	418.50	560.00	349.50	277.00
	350	**15**	**Mean**	287.55	564.75	237.45	369.30	483.35	308.15	271.65
6			**STD**	20.61613	149.55364	7.201273	19.70223	55.12664	22.95170	20.14537
			Min	255.50	383.00	228.50	339.00	386.50	268.50	248.50
			Max	312.50	910.00	247.50	401.00	583.00	339.00	312.50
	370	**16**	**Mean**	280.70	744.35	245.25	397.90	576.95	309.80	269.85
7			**STD**	24.91229	182.2534	7.32290	42.87242	64.74756	25.50620	13.38956
			Min	255.50	526.00	233.00	317.50	482.50	267.50	247.00
			Max	320.50	1012.50	258.00	459.50	675.50	341.00	287.50
	390	**16**	**Mean**	299.05	1019.75	247.85	482.55	636.30	304.20	276.05
8			**STD**	17.29716	225.13579	6.98430	24.78177	78.18681	26.66166	12.00335
			Min	272.00	704.50	239.00	439.50	536.50	257.00	261.50
			Max	319.50	1351.50	259.50	517.50	762.50	339.50	298.50
	410	**17**	**Mean**	316.60	1005.00	263.00	422.10	574.00	316.30	285.30
9			**STD**	26.81293	202.71175	10.77548	72.13059	61.44057	21.33880	16.946320
			Min	283.00	594.00	250.50	333.00	464.00	286.00	268.00
			Max	369.00	1289.00	280.00	522.00	670.00	356.50	324.00
10	**430**	**17**	**Mean**	305.35	974.95	251.00	406.00	578.45	314.05	277.70
			STD	25.20697	152.64309	10.93160	35.21284	89.40558	23.90542	14.75014
			Min	274.00	638.00	233.50	346.50	450.50	279.00	257.00
			Max	341.50	1218.50	268.50	457.50	764.50	356.50	298.00

Table 5.17 summarizes the hybrid algorithm test results for large problems. Accordingly, the proposed hybrid algorithm outperforms the modified algorithm in every instance and produces the best minimum values. Compared to other algorithms, FA produces the best minimum values for nine instances: 1, 3, 5, 6, 7, 8, 9, and 10. Conversely, when (N = 270, M = 13), the hybrid algorithm offers the best minimum value.

From a comparative perspective, the hybrid algorithm demonstrates superior performance, yielding significant results relative to other algorithms that may successfully solve the proposed problem. The results of the statistical tests suggest that the hybrid algorithm's performance is significantly different from the other six algorithms. However, it is noteworthy that the results of FA and SA demonstrate a significant competitive convergence performance of the proposed algorithm. Figure 5.7 provides the overall performance convergence of the proposed hybrid algorithm across all instances compared to other algorithms based on minimum value criteria. Remarkably, the proposed hybrid algorithm demonstrates superiority over the six algorithms in the small-sized problem.

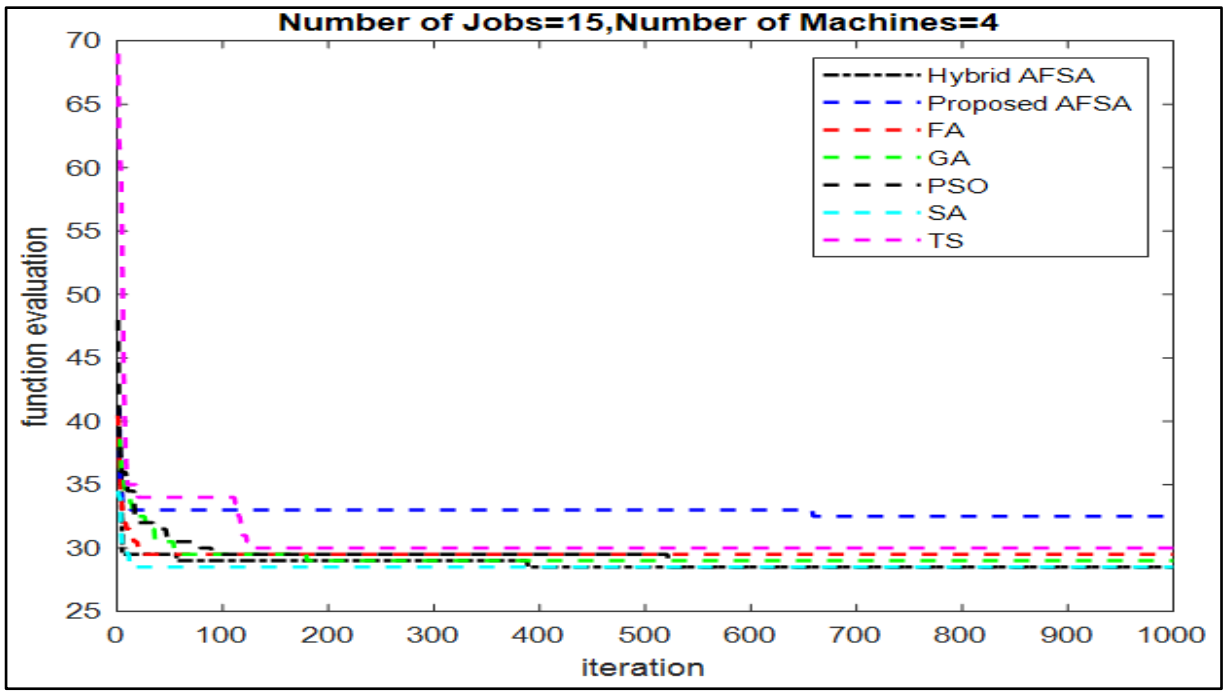


Figure 5.7. The convergence graph of the hybrid algorithm for small-sized problems

Figure 5.7 shows the graphs of hybrid AFSA and proposed AFSA relative to other common single metaheuristic algorithms (FA, GA, PSO, SA, TS) for a small-size problem instance (N = 15, M = 4), to identify which algorithms produce the highest solution quality and convergence speed. At iteration 0, all algorithms start with evaluations that are relatively high (between 50-70), indicating that their initial random solutions are not yet optimized. Within the first 50–100 iterations, almost all of the algorithms show a sharp initial decline in function evaluation. Hybrid AFSA (black dashed), FA (red dashed), GA (green dashed), PSO (solid black), and SA (cyan dashed) all converge very quickly and have relatively close solution behaviors in the same range of iterations.

TS (magenta dashed) exhibits a comparatively slower and more gradual convergence, particularly around iteration 150, while proposed AFSA (blue dashed) maintains a higher

function evaluation for a longer period of time before showing a notable decline around iteration 650.

At iteration 1000, the best evaluations are reached by hybrid AFSA, FA, GA, PSO, and SA, all of those converge to a value that is very close to the best, around 28–29. This indicates they are highly effective at determining great solutions for this particular problem. TS converges to a slightly higher value, around 30, while the proposed AFSA achieves the highest function evaluation among the compared algorithms, reaching around 31. Additionally, once converged, most algorithms showed stable performance.

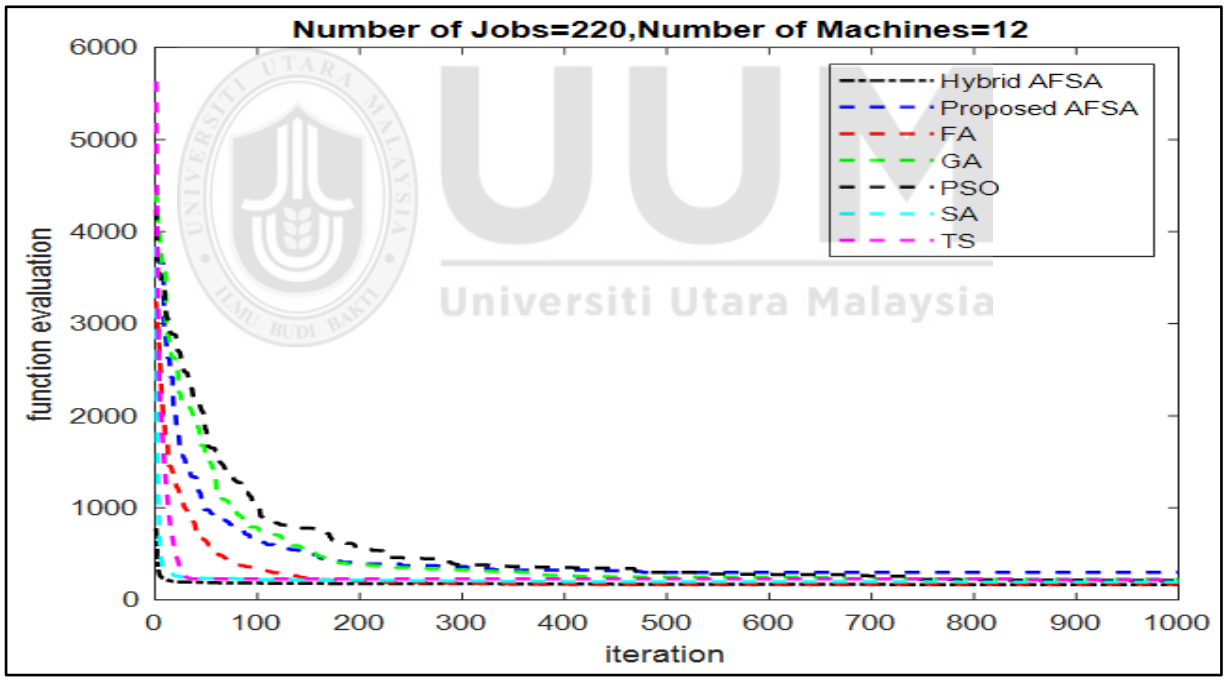


Figure 5.8. The convergence graph of the hybrid algorithm for medium-sized problem.

Figure 5.8 shows the convergence graph of hybrid AFSA for a medium-size problem instance (N = 220, M = 12). At iteration 0, algorithms start with high evaluations (around 5000-5500), indicating increasing problem complexity and the initial random solutions are

not the best. Within the first 50–100 iterations, all algorithms show a rapid decline in function evaluation, indicating their ability to rapidly improve initial solutions.

In 100–200 iterations, hybrid AFSA, FA, PSO, SA, and TS achieve fast convergence to their assigned solution. However, it takes more iterations for the proposed AFSA and GA to reach their finish line, showing more gradual initial convergence than the others.

From the first to the end of iterations 1000, hybrid AFSA clearly demonstrates the best performance with the lowest evaluation at 200–250. This is an impressive result that demonstrates how well it solves medium-sized problems. Additionally, FA, PSO, SA, TS, and GA all converge to very similar, competitive values that fall between around 250-300. Lastly, proposed AFSA performs worst and converges to a higher value, around 300–350.

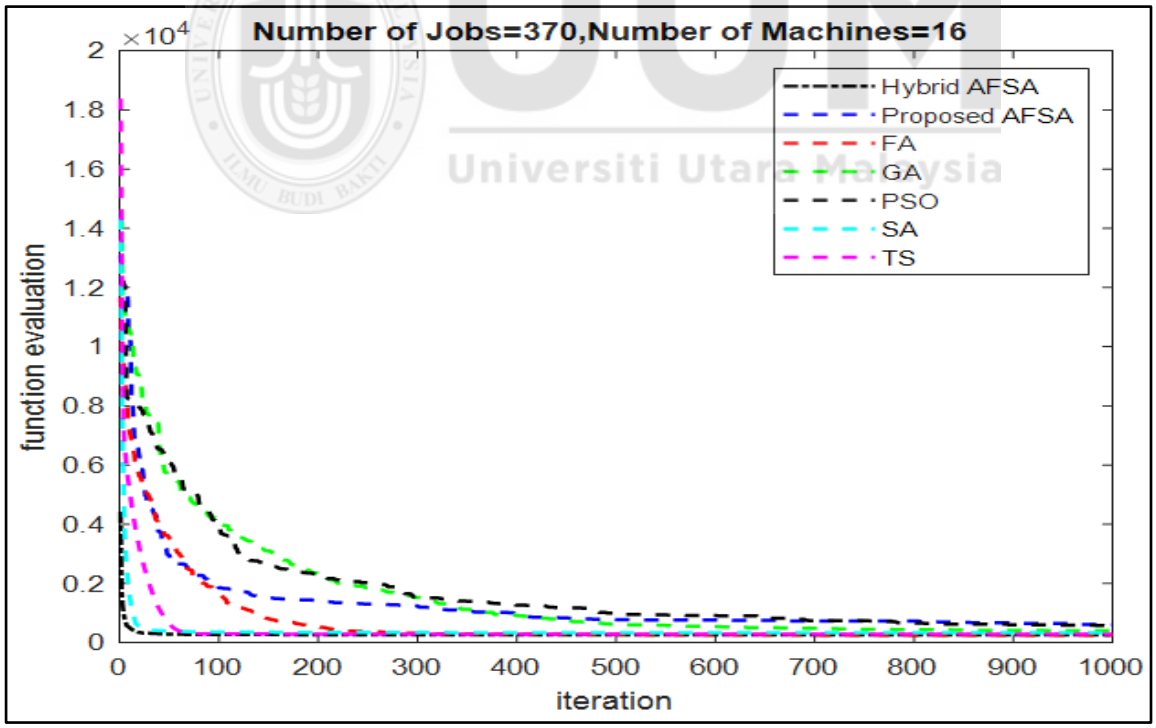


Figure 5.9. Illustrates the convergence graph of the hybrid algorithm for large-sized problem.

Figure 5.9 shows the convergence graph of hybrid AFSA for a large-size problem instance (N = 370, M = 16). To identify algorithms that will maintain superior solution quality and convergence when faced with a greatly expanded search space and increasingly complex computational problems. At iteration 0, all algorithms start with high evaluations (around 12,000-18,000), which is common for large, complex problems with initial solutions that are far from the best. Within the first 50-100 iterations, all algorithms show a rapid initial drop in evaluation, indicating that they quickly move away from extremely low solutions.

Hybrid AFSA and SA shows the fast convergence, with function evaluations dropping to very low levels almost immediately (within the first 50 iterations). This indicates they are very efficient at discovering promising areas of the search space. However, FA, Proposed AFSA, and TS converge relatively quickly. Meanwhile, GA and PSO exhibit slower convergence, requiring more iterations to achieve their best level.

At iteration 1000, FA is clearly the best performer, with the lowest evaluation, around 230-240. This is an excellent result for a large-scale problem. The second best is TS, which converges to a value between 240-250. Hybrid AFSA converges around 250-260. SA converges to approximately 260-270. GA, PSO, and the proposed AFSA all converge to a close higher value of around 300–530. Almost algorithms demonstrate good stability and robustness, leading to competitive solutions.

The computational results are summarized in Tables 5.18, 5.19, and 5.20 for the small, medium, and large-sized problems, respectively, in terms of the Wilcoxon test among the six algorithms. The best results amongst the comparative algorithms for each problem are presented in grey.

Table 5.18

Results of the Wilcoxon signed-rank test for small-sized problems

Problems		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Hybrid	***p*-value**	0.10900	0.04690	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	49.5	49.5	55	55	55	55	55	55	55	55
VS	**R-**	5.5	5.5	0	0	0	0	0	0	0	0
Proposed Modified	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	1.00000	1.00000	0.00195	0.20300	0.09380	0.44500	0.74000	0.27000	0.67600	0.71100
AFSA	**R+**	27	27	55	52	44.5	40	27	45	37	45
VS	**R-**	28	28	0	3	10.5	15	28	10	18	10
FA	**Ind.**	-1	-1	1	1	1	1	-1	1	1	1
Hybrid	***p*-value**	1.00000	1.00000	0.00391	0.00195	0.00391	0.00781	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	32	27	54.5	55	54.5	53	55	55	55	55
VS	**R-**	23	28	0.5	0	0.5	2	0	0	0	0
GA	**Ind.**	1	-1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	1.00000	1.00000	0.00781	0.12300	0.00781	0.00781	0.00391	0.00195	0.00391	0.00195
AFSA	**R+**	32	32	53.5	52	53.5	54	54	55	54	55
VS	**R-**	23	23	1.5	3	1.5	1	1	0	1	0
PSO	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.21875	0.03125	0.15625	0.01562	0.11132	0.41406	0.10937	0.41406	0.66015	0.17578
AFSA	**R+**	47	49.5	47	52	49	44.5	44.5	45	42.5	47
VS	**R-**	8	5.5	8	3	6	10.5	10.5	10	12.5	8
SA	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.37500	1.00000	0.00976	0.00781	0.00781	0.08203	0.91406	0.15234	0.20312	0.02539
AFSA	**R+**	44	27	54	53	53.5	50.5	44.5	49	45	54
VS	**R-**	11	28	1	2	1.5	4.5	10.5	6	10	1
TS	**Ind.**	1	-1	1	1	1	1	1	1	1	1

Table 5.19

Results of the Wilcoxon signed-rank test for medium-sized problems

Problems		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Hybrid	***p*-value**	0.00195	0.00195	0.01560	0.00195	0.00195	0.00391	0.00391	0.00195	0.00195	0.00195
AFSA	**R+**	55	54	52	55	55	54	54	55	55	55
VS	**R-**	0	1	3	0	0	1	1	0	0	0
Proposed Modified	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.30078	0.86700	0.00195	0.19300	0.01950	0.00195	0.01170	0.00195	0.00391	0.00391
AFSA	**R+**	47	45	0	34	19	0	14.5	0	10	10
VS	**R-**	8	10	55	21	36	55	40.5	55	45	45
FA	**Ind.**	1	1	-1	1	-1	-1	-1	-1	-1	-1
Hybrid	***p*-value**	0.00195	0.00195	0.01950	0.00195	0.00977	0.00977	0.00977	0.00195	0.01950	0.00391
AFSA	**R+**	55	55	52	55	52	54	52	55	52	54
VS	**R-**	0	0	3	0	3	1	3	0	3	1
GA	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00391	0.00977	0.00391	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	55	54	52	54	55	55	55	55
VS	**R-**	0	0	0	1	3	1	0	0	0	0
PSO	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.24609	0.49218	0.25000	0.55664	0.02734	0.91015	0.31250	1.00000	0.23242	0.92187
AFSA	**R+**	47	45	50.5	45	54	37	47	40	52	40
VS	**R-**	8	10	4.5	10	1	18	8	15	3	15
SA	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.04296	0.00976	0.82031	0.00585	1.00000	0.05273	0.55664	0.07812	0.10546	0.00585
AFSA	**R+**	50.5	52	42.5	54	40	27	40	34	27	10
VS	**R-**	4.5	3	12.5	1	15	28	15	21	28	45
TS	**Ind.**	1	1	1	1	1	-1	1	1	-1	-1

Table 5.20

Results of the Wilcoxon signed-rank test for large-sized problems

Problems		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	55	55	55	55	55	55	55	55
VS	**R-**	0	0	0	0	0	0	0	0	0	0
Proposed Modified	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.01171	0.00977	0.00195	0.00391	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	19	10	0	10	0	0	0	0	0	0
VS	**R-**	36	45	55	45	55	55	55	55	55	55
FA	**Ind.**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Hybrid	***p*-value**	0.00585	0.00390	0.00195	0.00977	0.00195	0.00195	0.00195	0.00195	0.00586	0.00195
AFSA	**R+**	54	54	55	54	55	55	55	55	54	55
VS	**R-**	1	1	0	1	0	0	0	0	1	0
GA	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.00195	0.00195	0.00391	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	54	55	55	55	55	55	55	55
VS	**R-**	0	0	1	0	0	0	0	0	0	0
PSO	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.67773	0.01367	0.12500	0.32226	1.00000	0.03710	0.03710	0.55664	0.49218	0.55664
AFSA	**R+**	45	52	49	49	40	52	52	49	49	45
VS	**R-**	10	3	6	6	15	3	3	6	6	10
SA	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.78515	0.94531	0.20117	0.13085	0.23242	0.22265	0.43164	0.00195	0.01757	0.03710
AFSA	**R+**	45	45	34	27	40	27	40	0	10	19
VS	**R-**	10	10	21	28	15	28	15	55	45	36
TS	**Ind.**	1	1	1	-1	1	-1	1	-1	-1	-1

According to the Wilcoxon signed-ranks test tables, the results of the proposed hybrid AFSA algorithm suggest that, in each pairwise comparison, the hybrid algorithm performs significantly better than the proposed modified algorithm, PSO, and SA for small, medium, and large-sized problems, where *p*-values are often less than 0.05 and rank sum values R+ in almost all cases which confirm statistically significant differences and strongly favor the hybrid algorithm. Additionally, FA differs significantly from single metaheuristic algorithms in the 1, 2, and 7 instances of the small-sized problem, the 3, 5, 6, 7, 8, 9, and 10 instances of the medium-sized problem, and all ten instances of the large-sized problem. This indicates that the competitiveness of FA is confirmed by many *p*-values > 0.05 and rank sum values R- in almost all problems.

In each pairwise comparison, GA also shows a important difference in algorithm performance for small, medium, and large-sized problems, except for the second instance for small-sized problems, is a result of many *p*-values very small except for problems 1 and 2, where a strong significant difference was observed and the hybrid dominates better. Finally, the proposed hybrid algorithm performs better than the TS, except for the second instance for small-sized problems. This includes 6, 9, and 10 instances for the medium-sized problem and 4, 6, 8, 9, and 10 instances for the large-sized problem, which was derived from rank R+ values and *p*-values that consistently favored the hybrid algorithm; in most cases, it makes a significant difference, usually outperforming but sometimes competing with TS.

5.4.3 Computational Results of Hybrid Metaheuristics Algorithms

Finally, a third set of tests was performed to compare the performance of the proposed hybrid AFSA algorithm with three existing hybrid algorithms: the Hybrid Artificial Bee Colony Algorithm (HABC) introduced by Lin and Ying (2014), the Enhanced Salp Swarm Algorithm based on FA (SSAFA) introduced by Ewees et al. (2021), and Whale Optimization Algorithm and FA (WOAFA) introduced by Al-qaness et al. (2021). The three hybrid algorithms developed aim to minimize the objective of makespan, considering machine-dependent and job sequence-dependent setup times to address the UPMSP. All those algorithms utilize test problem instances to identify optimal solutions and evaluate their performance, demonstrating significantly improved results.

Tables 5.21, 5.22, and 5.23 describe the results of small, medium, and large-sized problems, respectively. These results aim to evaluate the hybrid algorithm's efficiency based on other hybrid algorithms using the same four performance evaluation criteria.

Table 5.21

The computational results of the proposed hybrid AFSA with three hybrid algorithms for small-sized problems

	N	M		Hybrid AFSA	SSAFA	WOAFA	HABC
1	**5**	**3**	**Mean**	15.50	17.45	17.20	15.50
			STD	0	1.09163	1.05934	0
			Min	15.50	16.00	15.50	15.50
			Max	15.50	19.00	18.50	15.50
2	**10**	**3**	**Mean**	28.30	37.65	38.00	28.30
			STD	0.25819	3.98608	2.46136	0.25819
			Min	28.00	34.00	33.00	28.00
			Max	28.50	41.50	43.50	28.50
3	**15**	**4**	**Mean**	29.95	43.00	43.95	32.25
			STD	0.28382	4.26874	4.64548	0.58925
			Min	29.50	34.50	37.00	31.00
			Max	30.50	48.00	53.00	33.00
4	**20**	**4**	**Mean**	37.50	60.35	59.70	42.65
			STD	0	3.60593	4.9340	0.94428
			Min	37.50	54.00	53.00	41.00
			Max	37.50	66.50	70.50	44.00
5	**25**	**5**	**Mean**	37.45	65.95	63.90	46.00
			STD	1.23490	5.63939	7.70209	1.22474
			Min	36.50	58.00	53.00	44.00
			Max	40.00	76.50	77.00	48.00
6	**30**	**5**	**Mean**	51.00	94.10	99.40	61.40
			STD	2.10818	7.72010	12.04574	1.62958
			Min	48.00	82.50	89.00	60.00
			Max	53.50	106.00	127.50	65.50
7	**35**	**6**	**Mean**	42.80	80.70	81.10	55.90
			STD	1.87379	5.90291	11.18232	1.30809

			Min	41.50	74.00	70.00	53.50
			Max	47.00	89.50	103.00	57.50
8	**40**	**6**	**Mean**	54.30	126.65	129.00	71.00
			STD	2.45175	14.31015	14.84737	1.76383
			Min	50.00	96.00	101.00	68.00
			Max	57.00	147.50	145.00	74.00
9	**45**	**7**	**Mean**	53.00	149.10	130.15	73.90
			STD	2.47206	19.20763	29.49298	3.08940
			Min	50.00	114.50	91.50	68.50
			Max	58.50	178.00	181.50	77.50
10	**50**	**7**	**Mean**	54.45	122.65	113.30	74.70
			STD	2.31480	16.20365	15.75718	1.88856
			Min	51.50	101.50	90.00	72.00
			Max	58.50	150.00	136.50	78.00

Table 5.22

The computational results of the proposed hybrid AFSA with three hybrid algorithms for medium-sized problems

	N	**M**		**Hybrid AFSA**	**SSAFA**	**WOAFA**	**HABC**
1	**60**	**8**	**Mean**	59.25	202.35	209.55	94.55
			STD	1.08653	48.64500	34.42015	3.71520
			Min	58.00	148.50	170.50	87.00
			Max	61.50	325.50	283.50	98.50
2	**80**	**8**	**Mean**	80.15	399.85	413.75	172.60
			STD	5.70599	73.78800	59.83832	12.64427
			Min	74.50	324.00	302.50	154.50
			Max	93.50	551.50	495.50	187.00
3	**100**	**9**	**Mean**	106.45	781.05	894.65	369.80
			STD	8.37804	102.96100	149.48096	44.57091

			Min	91.00	639.00	595.00	260.00
			Max	118.50	967.00	1067.00	409.00
4	**120**	**9**	**Mean**	116.20	981.40	999.10	469.50
			STD	9.22315	65.97718	222.74247	30.20853
			Min	108.00	881.00	677.00	412.00
			Max	136.50	1100.00	1342.50	532.00
5	**140**	**10**	**Mean**	128.75	1413.20	1494.65	702.30
			STD	12.54824	277.55211	234.49390	47.96711
			Min	114.50	1075.00	1139.00	643.50
			Max	149.00	1944.50	1946.00	810.00
6	**160**	**10**	**Mean**	152.75	1974.25	2155.50	1006.05
			STD	19.08497	383.51396	330.15249	62.02304
			Min	135.00	1248.00	1440.00	904.00
			Max	201.00	2413.00	2545.50	1089.50
7	**180**	**11**	**Mean**	176.80	3004.90	3340.00	1600.90
			STD	21.67461	359.29798	297.0637	58.54665
			Min	142.50	2544.50	2856.00	1486.00
			Max	204.00	3590.50	3697.00	1672.00
8	**200**	**11**	**Mean**	219.35	4025.90	3995.40	2279.95
			STD	16.65674	702.55679	395.99514	98.72026
			Min	195.50	2909.50	3234.00	2072.50
			Max	241.50	5425.50	4667.50	2419.50
9	**220**	**12**	**Mean**	193.65	4353.75	4136.75	2242.65
			STD	16.47059	514.28198	595.72207	192.52475
			Min	165.00	3494.50	2959.50	1810.00
			Max	218.00	4941.00	4802.50	2500.50
10	**240**	**12**	**Mean**	215.75	5888.35	5706.00	3172.20
			STD	38.46156	807.91470	895.36590	72.26655
			Min	179.00	5050.50	4814.50	3092.50
			Max	299.00	7146.00	7452.00	3307.50

Table 5.23

The computational results of the proposed hybrid AFSA with three hybrid algorithms for large-sized problems

	N	M		Hybrid AFSA	SSAFA	WOAFA	HABC
1	**250**	**13**	**Mean**	212.30	4468.45	4871.85	2530.65
			STD	22.27380	581.90680	734.37229	109.02116
			Min	181.00	3650.50	3849.50	2334.50
			Max	249.00	5397.00	6067.00	2643.50
2	**270**	**13**	**Mean**	245.55	6829.65	7130.40	3850.65
			STD	25.83112	939.38446	1334.66293	149.32478
			Min	205.00	5263.50	5236.00	3597.00
			Max	291.50	8086.00	9715.50	4022.00
3	**290**	**14**	**Mean**	241.55	6944.05	7494.50	3975.40
			STD	34.13083	1272.97114	753.21927	254.50559
			Min	177.00	4614.00	6281.50	3583.00
			Max	282.50	9618.50	8659.50	4341.00
4	**310**	**14**	**Mean**	257.20	9536.15	10018.80	5859.80
			STD	38.92528	1303.10763	1257.90494	253.25549
			Min	194.50	6855.50	7934.00	5551.50
			Max	313.50	11561.00	11671.00	6249.50
5	**330**	**15**	**Mean**	253.90	9190.90	9357.15	5398.80
			STD	36.65211	1540.79100	783.08432	147.39350
			Min	219.00	6551.50	7979.00	5165.00
			Max	333.00	11610.00	10406.50	5662.50
6	**350**	**15**	**Mean**	264.20	12498.30	12691.55	7569.80
			STD	39.23377	1723.84667	1500.12526	498.54277
			Min	225.50	10130.50	10683.50	6756.50
			Max	322.00	15622.00	14962.00	8159.50
7	**370**	**16**	**Mean**	272.40	13146.50	12460.20	7640.55

			STD	17.39699	1889.99738	1183.08291	238.26461
			Min	241.50	9930.00	10021.50	7326.50
			Max	299.00	16044.50	14032.00	8008.50
8	**390**	**16**	**Mean**	288.60	13775.25	14127.90	8443.25
			STD	33.45710	1710.99555	1789.86586	504.40031
			Min	238.50	11030.00	10915.00	7376.50
			Max	347.50	16607.00	17017.50	9128.00
9	**410**	**17**	**Mean**	317.45	17341.70	17171.35	10385.65
			STD	27.20237	1885.78464	1477.01476	415.26364
			Min	278.00	14364.00	15022.00	9736.50
			Max	371.00	20119.00	19074.50	11041.00
10	**430**	**17**	**Mean**	302.95	17876.65	17598.20	10697.35
			STD	16.96638	2490.35238	2236.56825	465.44011
			Min	275.00	13378.50	13192.00	9877.00
			Max	331.00	21783.50	20180.50	11369.50

The results in Tables 5.21, 5.22, and 23 illustrate the mean, STD, Min, and Max values, respectively. The results indicate that the proposed hybrid AFSA algorithm outperforms the other hybrid algorithms across all problem sizes based on the best minimum values achieved. Furthermore, from the results of STD, it can be concluded that the proposed hybrid AFSA algorithm is more stable than others, followed by HABC, WOAFA, and SSAFA, which ranked second, third, and fourth, respectively. Notably, the high performance of the developed hybrid AFSA algorithm is the result of combining the advantages of the proposed modified AFSA and the neighborhood search algorithm. The modified AFSA and neighborhood search algorithm update solutions competitively based on their quality.

Furthermore, Figures 5.10, 5.11, and 5.12 highlight the entire convergence of the proposed hybrid algorithm through all instances compared with other hybrid algorithms according to minimum value criteria. As can be seen, the proposed hybrid algorithm exhibits significant superiority over the three algorithms across all problem sizes.

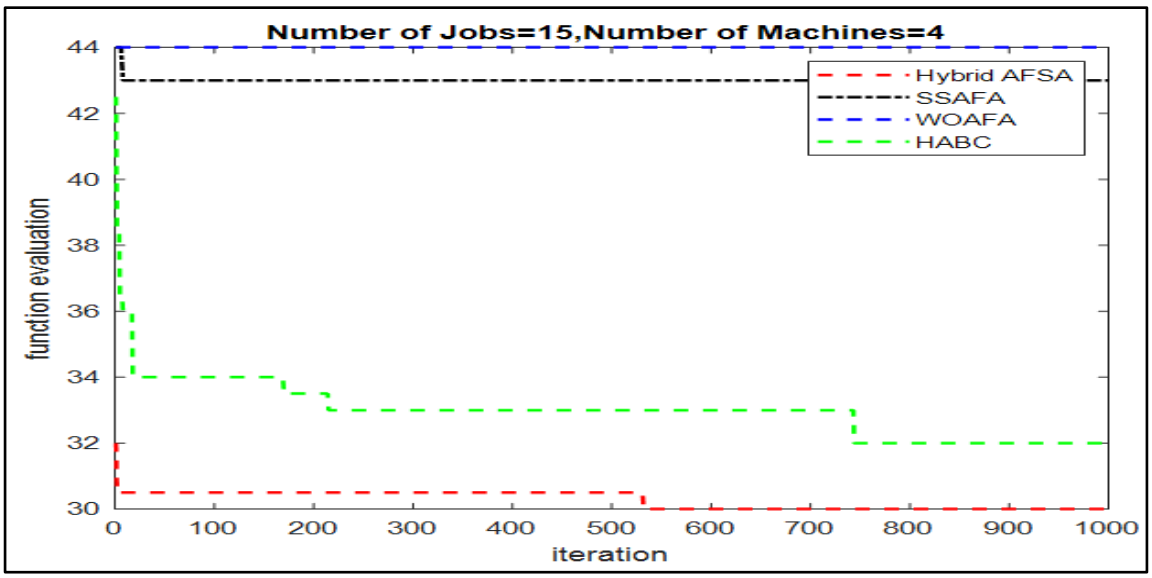


Figure 5.10. The performance of hybrid AFSA, HABC, WOAFA, and SSAFA for small-sized problem.

Figure 5.10 shows the convergence graph of the proposed hybrid AFSA with other three hybrid algorithms to evaluate their solution quality and convergence speed for a small-sized problem instance ($N = 15$, $M = 4$). The function evaluation values listed range from 30-44, indicating an emphasis on subtle differences in solution quality. At iteration 0, the initial evaluations differ significantly. WOAFA (dashed blue line) starts high (around 44), followed by SSAFA (dashed black line) at 43, HABC (green dashed) at 42, and hybrid AFSA (red dashed) at 32.

Within the first few iterations, the hybrid AFSA converges quickly, falling from 32 to 30.5 and then maintaining that value until around 550 iterations. HABC also converges quickly, dropping from 42 to 34 within the first 50 iterations and then again to around 32 at iteration 750. There is almost no convergence between WOAFA and SSAFA, as they remain at high initial evaluation values for the entire 1000 iterations.

At iteration 1000, the hybrid AFSA proved to be the best performer, achieving the lowest evaluation value of 30, which was reached at iteration 600. The algorithm demonstrated excellent stability by rapidly identifying a high-quality solution and maintaining it throughout the remaining iterations. The second-best is HABC, which converges to about 32 and showing some stability after an initial drop, with further improvement later in the iterations. Meanwhile, SSAFA and WOAFA perform badly, and failing to optimize the problem in this particular instance.

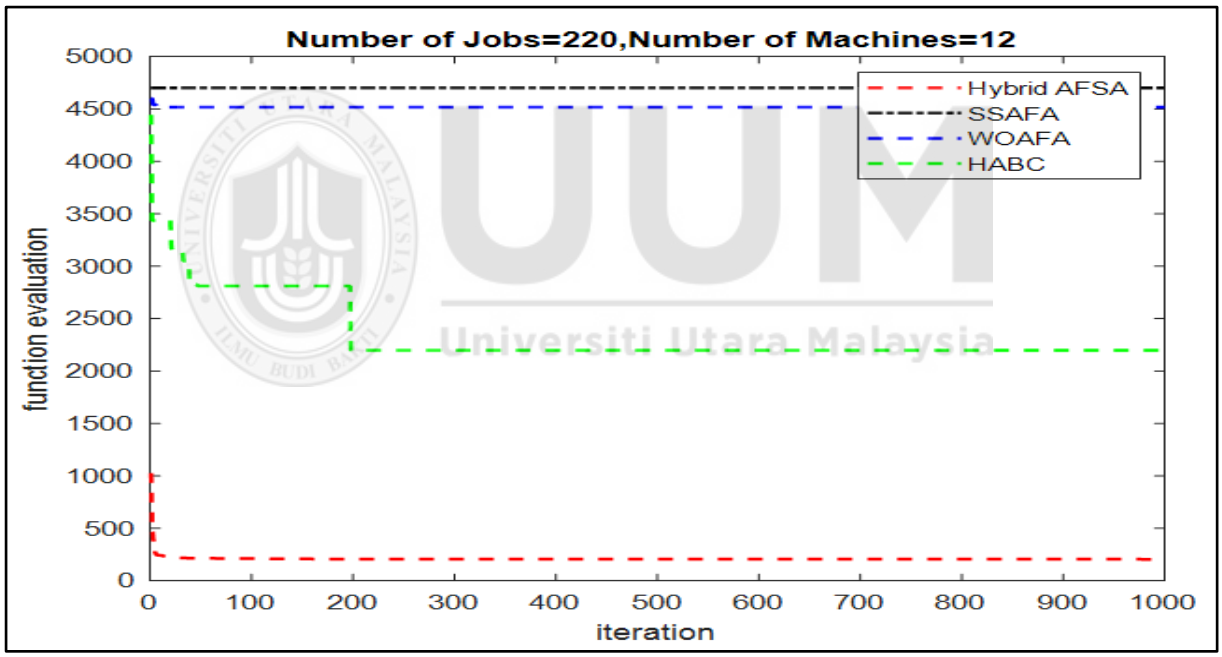


Figure 5.11. The performance of hybrid AFSA, HABC, WOAFA, and SSAFA for medium-sized problem.

Figure 5.11 shows the convergence graph of the proposed hybrid AFSA for a medium-sized problem instance ($N = 220$, $M = 12$). At iteration 0, SSAFA and WOAFA begin with

high function evaluations (around 4500-4700), indicating insufficient solutions for the problem size. HABC starts at around 4000, which is also high.

Hybrid AFSA begins at around 1000, which is significantly lower than the others, indicating a more effective initial search phase. In the first 20 to 30 iterations, the hybrid AFSA shows extremely rapid convergence, dropping from 1000 to 200 and then staying at that low value until the end of the iterations. In the first 50 iterations, HABC shows a gradual convergence, falling from 4000 to 2800, then to 2200 around iteration 200, and finally settling around 2200. Meanwhile, SSAFA and WOAFA have almost no convergence. They stay at their high initial evaluation values throughout the 1000 iterations.

At iteration 1000, hybrid AFSA is the overall best performer, converging to the lowest evaluation, which is around 200, and demonstrating excellent stability and robustness, quickly finding a high-quality solution and maintaining it throughout the iteration. The result is an excellent solution for a medium-sized problem. HABC is second best, with a value of around 2200, showing some stability after a gradual decline.

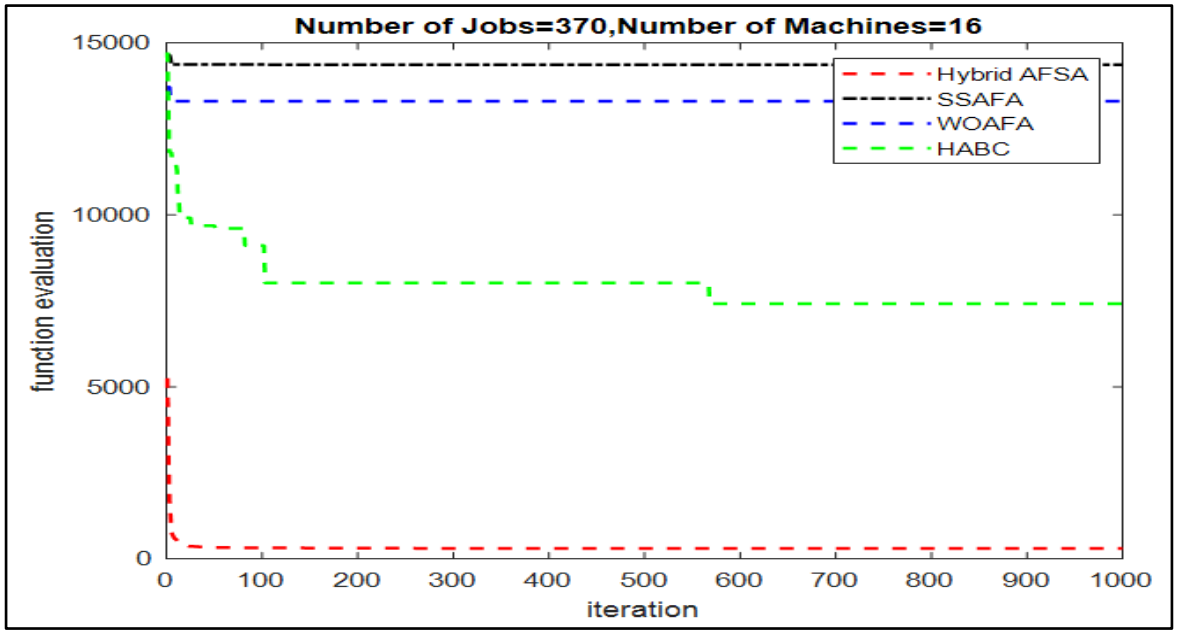


Figure 5.12. The performance of hybrid AFSA, HABC, WOAFA, and SSAFA for large-sized problem.

Figure 5.12 shows the convergence graph of the proposed hybrid AFSA for a large-sized problem instance (N = 370, M = 16). The range of function evaluation values from 0 to 15,000 is significantly higher than the previous graphs, indicating the increased problem complexity and size. At iteration 0, SSAFA and WOAFA have high function evaluations (around 14,000-14,500), indicating poor initial solutions and lack of convergence for this problem size. They remain at their very high initial evaluation values throughout the 1000 iterations, showing a complete failure to optimize the problem in this particular instance. HABC starts around 12,500, while hybrid AFSA starts around 5,000, which is much lower than the others, indicating a very effective initial search phase. In the first 20-30 iterations, the hybrid AFSA drops from 5000 to less than 500 and then maintains that low value, demonstrating remarkably speed convergence. Gradual convergence is demonstrated by

HABC, which decreases from 12,500 to 9,500 in the first 50 iterations, then to 7,500 around iteration 150, and finally to around 7,000.

At iteration 1000, hybrid AFSA achieved the best overall performance, with the lowest evaluation around 200-300. This is an excellent result for a large-sized problem. As a result, it exhibits outstanding stability and robustness by quickly identifying a high-quality solution and maintaining it throughout the iterations. HABC is the second best, reaching a value of around 7,000 and showing some stability after its gradual drops.

In terms of the Wilcoxon test between the three hybrid algorithms, the computational results are summarized in Tables 5.24, 5.25, and 5.26 for the small, medium, and large-sized problems, respectively. The results of the proposed hybrid AFSA algorithm prove that, in each pairwise comparison, the hybrid algorithm noticeably exhibits a significant difference between the three hybrid algorithms SSAFA, WOAFA, and HABC. Strong statistical significance is confirmed by the results, which show p-values = 0.001953 (< 0.05) for every comparison. When combined with R+ = 55 and R- = 0, this indicates that Hybrid AFSA totally outperforms AFSA, WOAFA, and HABC.

Table 5.24

Results of the Wilcoxon signed-rank test for small-sized problems

Problems		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	55	55	55	55	55	55	55	55
VS	**R-**	0	0	0	0	0	0	0	0	0	0
SSAFA	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.00390	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	54.5	55	55	55	55	55	55	55	55	55
VS	**R-**	0.5	0	0	0	0	0	0	0	0	0
WOAFA	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	1.00000	1.00000	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	27.5	35.5	55	55	55	55	55	55	55	55
VS	**R-**	27.5	19.5	0	0	0	0	0	0	0	0
HABC	**Ind.**	0	1	1	1	1	1	1	1	1	1

Table 5.25

Results of the Wilcoxon signed-rank test for small-sized problems

Problems		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	55	55	55	55	55	55	55	55
VS	**R-**	0	0	0	0	0	0	0	0	0	0
SSAFA	**Ind.**	1	1	1	1	1	1	1	1	1	1
	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195

Hybrid	**R+**	55	55	55	55	55	55	55	55	55	55
AFSA	**R-**	0	0	0	0	0	0	0	0	0	0
VS	**Ind.**										
WOAFA		1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	55	55	55	55	55	55	55	55
VS	**R-**	0	0	0	0	0	0	0	0	0	0
HABC	**Ind.**	1	1	1	1	1	1	1	1	1	1

Table 5.26

Results of the Wilcoxon signed-rank test for large-sized problems

Problems		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	55	55	55	55	55	55	55	55
VS	**R-**	0	0	0	0	0	0	0	0	0	0
SSAFA	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	55	55	55	55	55	55	55	55
VS	**R-**	0	0	0	0	0	0	0	0	0	0
WOAFA	**Ind.**	1	1	1	1	1	1	1	1	1	1
Hybrid	***p*-value**	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195	0.00195
AFSA	**R+**	55	55	55	55	55	55	55	55	55	55
VS	**R-**	0	0	0	0	0	0	0	0	0	0
HABC	**Ind.**	1	1	1	1	1	1	1	1	1	1

Figure 5.13 shows a performance chart that summarizes the comparison results across ten instances of standard AFSA, modified AFSA, and hybrid AFSA algorithms based on the best minimum values for larg-size problems. These results demonstrate that the hybrid AFSA algorithm is superior to the other algorithms.

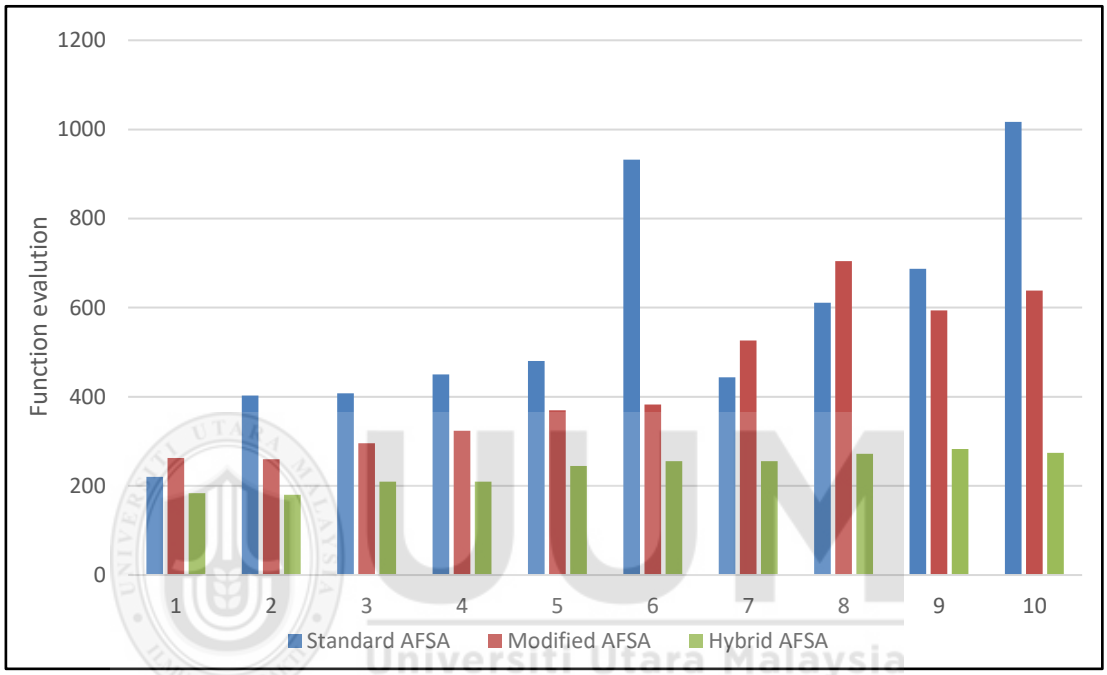


Figure 5.13. Comparison between the standard AFSA, Modified AFSA, Hybrid AFSA algorithms

The chart compares the efficiency of three proposed algorithms, all based on the AFSA, and shows that the hybrid AFSA (green bars) clearly performs best, indicating its superior ability to converge quickly and efficiently on the solution for all tested problems. Meanwhile, the standard AFSA (blue bars) is the least efficient and has the greatest degree of performance variability. Then, in most situations, the Modified AFSA (red bars) outperforms the standard AFSA; however, in particular instances, its performance is less effective than that of the hybrid AFSA.

5.5 Chapter Summary

This chapter provides the computational results and discussion of the proposed modified and hybrid AFSA algorithms to minimize the multi-objective makespan and total tardiness. The performance of the proposed algorithms in identifying the best solutions for UPMSP is evaluated across three sets of problem sizes, each with ten randomly generated instances. The small problems have up to 50 jobs and up to seven machines, while the medium and large-sized problems have up to 240 jobs and 12 machines and up to 430 jobs and 17 machines, respectively.

The comparison results of the modified AFSA prove that the proposed algorithm is more competitive than the standard AFSA and five modified algorithms. Specifically, it improves the exploitation capability of the AFSA algorithm. It also demonstrates superior performance against the AFSA and other modified AFSA algorithms in identifying the best solutions for almost all test problems.

The proposed hybrid AFSA algorithm has been compared with three sets to assess its effectiveness. The first test used five versions of modified comparison algorithms and standard AFSA, demonstrating that the proposed hybrid algorithm outperforms all evaluated comparative algorithms. Meanwhile, the second comparison test was evaluated against five single metaheuristic algorithms (FA, GA, PSO, SA, and TS) and the proposed modified algorithm. The hybrid algorithm demonstrates increased efficiency, significantly improving and effectively solving the proposed problem. At the same time, statistical tests suggest that the hybrid algorithm outperforms the proposed modified algorithm, GA, PSO, and TS in all test size problems. On the other hand, the results of FA and SA present a highly competitive convergence performance.

Moreover, the third comparison test was conducted against three existing hybrid algorithms, SSAFA, WOAFA, and HABC, which were used to minimize the proposed scheduling problems to solve UPMSP. The results reveal that the proposed hybrid AFSA algorithm outperforms the other hybrid algorithms across all problem sizes in terms of the best minimum values achieved.

The evaluation from the comparison indicates that the developed hybrid AFSA algorithm performs well since it combines the advantages of the proposed modified AFSA and the neighborhood search algorithm. In essence, modified AFSA and neighborhood search algorithms compete by updating solutions based on their quality, successfully producing good solutions.

CHAPTER SIX
CONCLUSIONS

6.1 Introduction

This chapter summarizes the research, highlighting the achievement of research objectives, key contributions, limitations, and possible directions for future research. Section 6.2 presents a detailed description of the research methodology employed. Consequently, Section 6.3 discusses the achievement of research objectives. Section 6.4 outlines the research contribution, while Section 6.5 outlines the limitations of the research. Future research suggestions are outlined in Section 6.6.

6.2 Summary of Unrelated Parallel Machine Scheduling Problem

This research proposes two new algorithms for solving existing linear multi-objective Unrelated Parallel Machine Scheduling Problems (UPMSPs) to minimize multi-objective makespan and total tardiness after extending and embedding triangular fuzzy parameters.

The first algorithm, the proposed modified Artificial Fish Swarm Algorithm (AFSA), improves the performance of the standard AFSA using three modification steps. First, aspiration behavior was incorporated into AFSA behaviors to increase effectiveness. Second, improved parameters such as step and visual were used to enhance equilibrium of global search ability and convergence rate. Finally, a transformation method was introduced to ensure that the algorithm is applicable for discrete optimization problems, such as UPMSP. The proposed modified algorithm was compared with AFSA and five modified versions of AFSA to validate and evaluate the effectiveness of the algorithm by performing three different sizes of problems.

Subsequently, the Wilcoxon signed-rank test was used to assess the algorithm's performance statistically. The findings show that the proposed algorithm performed noticeably more effectively than the others, particularly for medium and large-sized problems.

The second algorithm, the proposed hybrid AFSA, improves the performance of the proposed modified algorithm to get more precise results. This is achieved by integrating the proposed modified AFSA with the neighborhood search algorithm and using three neighborhood operations: swap, reversion, and insertion. Accordingly, the hybrid algorithm was compared with three sets of tests to verify the algorithm's effectiveness with the same three problems of different sizes. Furthermore, the Wilcoxon signed-rank test across all problem sizes was used to statistically assess the algorithm's performance with all comparative algorithms. First, it was conducted using AFSA and five modified versions of AFSA, followed by the proposed modified algorithm and five single metaheuristics algorithms, and finally, the existing three hybrid algorithms, SSAFA, WOAFA, and HABC. The results of the proposed hybrid AFSA revealed the best convergence to the optimal solution, outperforming the results of all comparative algorithms.

6.3 Achievement of Research Objectives

The main research objective and all four specific objectives were successfully achieved to develop an effective modified and hybrid AFSA algorithm capable of handling multi-objective UPMSP with fuzzy set theory. The main objective was achieved when the four specific objectives were successfully attained.

Four research questions were considered in Section 1.7, and the four corresponding research objectives that addressed these questions are detailed in Section 1.8. The main objective of the research was to develop a modified and hybrid AFSA algorithm capable of minimizing fuzzy multi-objective makespan and total tardiness in UPMSP and achieving the best possible results.

The first objective is formulating a Mixed Integer Linear Programming (MILP) model for UPMSP. It aims to minimize multi-objective makespan and total tardiness by extending Kongsri and Buddhakulsomsiri's (2020) linear multi-objective mathematical model and embedding Triangular Fuzzy Numbers (TFNs) of its main parameters, such as processing times and due dates. This formula can be employed to evaluate the proposed algorithms by solving the minimization problem and looking into the best possible solutions for the proposed problem. This objective is achieved in Chapter 3.

A literature review of various Machine Scheduling Problems (MSPs) with fuzzy environments involving distinct fuzzy numbers from different aspects of objectives and solving methods is provided to understand the current research. Therefore, the new modified fish swarm algorithm is designed to solve the proposed scheduling problem. Subsequently, the second objective is to develop a modified AFSA algorithm as a main contribution to increase the effectiveness of standard AFSA. The modification process comprises three aspects, adding a new behavior called aspiration behavior to the four main behaviors. It is inspired by the best position of Artificial Fish (AF), which was discovered and recorded on the bulletin board after performing all of the AFSA's behaviors and making it suitable for dealing with the proposed mathematical model.

Consequently, improved adaptive visual and step parameters should be adopted in the specific formulas used by Tan and Mohamad-Saleh (2020) instead of fixed parameters. Finally, the AFSA algorithm should be suitable for solving discrete UPMSPs and improving the original algorithm's search capabilities. This can be achieved using transformation method to convert the continuous solution into a discrete solution. The proposed modified algorithm satisfies the objective in Section 4.2.

The third objective is to develop a hybrid algorithm combining the proposed modified AFSA algorithm with a neighborhood search algorithm using three neighborhood operations to fully exploit the search space and identify the best solutions to achieve more precise solutions. The proposed hybrid algorithm achieves this objective in Section 4.3.

Wrapping up this research, the fourth objective evaluates the efficiency of the proposed algorithms using computational tests across different problem sizes. It utilizes statistical measures, including mean, Standard Deviation (STD), Minimum (Min), and Maximum (Max) values, in addition to the Wilcoxon signed-rank statistical test as criteria for performance evaluation. The modified algorithm was evaluated using the standard AFSA and five modified versions of AFSA. Correspondingly, the hybrid algorithm is evaluated with three sets of comparative algorithm tests to assess its performance and efficiency. The objective was achieved through a comparison study detailed in Sections 5.3 and 5.4.

Overall, the proposed algorithms of modified AFSA and its hybrid model improve the standard AFSA algorithm and provide an opportunity to improve its results. The objective of improving the AFSA algorithm's performance was a great success. In

solving the UPMSP model multi-objectively, the results of the two proposed algorithms demonstrate that they perform better than the most well-known algorithms.

6.4 Research Contributions

As a result of the research objectives, this study makes four distinct contributions. First, this research successfully developed an extended Mathematical Programming (MP) model based on the work of Kongsri and Buddhakulsomsiri (2020) to address a multi-objective UPMSP without time constraints. The model aims to minimize makespan and total tardiness within a fuzzy environment. Moreover, TFNs were employed to represent processing times and due dates to better capture uncertainty in scheduling, enhancing the model's ability to oversee real-world variability.

Secondly, this research critically examined the weaknesses of the standard AFSA and explored its effectiveness in solving UPMSP. The study identified key limitations of AFSA and proposed a new modified AFSA with three key modifications to enhance its performance in solving fuzzy multi-objective UPMSP. The objective is to minimize makespan and total tardiness in a fuzzy environment. The efficiency of the proposed modified AFSA is assessed by comparing its performance with the standard AFSA and various other modified versions of the algorithm. Correspondingly, the comparative analysis provides insights into the improvements achieved through the proposed modifications, demonstrating its potential for more efficient scheduling optimization.

Lastly, a new hybrid AFSA is proposed by integrating the modified AFSA with a neighborhood search algorithm to improve the proposed modified AFSA and achieve more precise solutions. This hybrid algorithm solves the proposed fuzzy multi-

objective UPMSP by minimizing makespan and total tardiness. The performance of the hybrid AFSA is assessed by a comprehensive comparison with three sets of comparative algorithms. The first set includes five different versions of modified AFSA algorithms, the second set comprises the proposed modified AFSA along with five single metaheuristic algorithms, and the final set consists of three existing hybrid algorithms. In particular, this comparison study intends to evaluate the efficacy of the hybrid AFSA in achieving improved scheduling optimization outcomes.

The outcomes of this research improve job scheduling and boost the productivity of UPMSP. This will improve scheduling quality across various industries, from manufacturing to services.

6.5 Limitations of the Study

This research concentrates on developing two new algorithms, modified and hybrid AFSA, to minimize the multi-objective function concerning UPMSP makespan and total tardiness within a fuzzy environment. Notably, the proposed modified AFSA algorithm concentrates on three aspects that can be adapted for the discrete MP model of UPMSP. It consistently exhibits the highest overall performance. Nonetheless, for some instances of testing involving several local optima, the efficacy of the proposed modified AFSA algorithm is unsatisfactory. Therefore, the hybrid AFSA algorithm was developed by integrating the modified AFSA with the neighborhood search algorithm to attain more accurate solutions through three neighborhood operations. This research compares the proposed algorithms to existing ones by simulating computational experiences and generating various sets of jobs and machines to assess the effectiveness and efficiency of the proposed algorithms. This research, however,

does not use benchmark optimization functions to assess the algorithm's performance. Furthermore, the proposed algorithms have not yet been applied to real-world case studies.

6.6 Future Research

The currently proposed algorithms can be further enhanced in machine scheduling environments by integrating them with other metaheuristic techniques to address a wider range of optimization objectives. For example, future research could focus on minimizing early and tardy penalties, weighted completion time, or weighted tardiness, expanding the applicability of the proposed approaches, and by including multi-objective performance evaluation using pareto-based metrics, a more comprehensive assessment of algorithm performance can be provided. Additionally, extending this study to incorporate more than two objectives could provide deeper insights into Multi-Objective Optimization (MOO) in scheduling.

Another promising direction is the inclusion of setup time constraints, represented using Trapezoidal Fuzzy Numbers (TrFNs), to refine the fuzzy multi-objective UPMSP model. This expanded model could be solved using alternative metaheuristic algorithms, such as the Firefly Algorithm (FA), to explore their effectiveness in overseeing scheduling uncertainties.

Furthermore, future research could apply the proposed algorithms to minimize multi-objective makespan and total tardiness in environments with more complex scheduling, like job shop or flow shop scheduling. Hence, investigating these scenarios would contribute to a broader understanding of how hybrid metaheuristic approaches

perform across various scheduling challenges. Additionally, this research can be expanded to include and evaluate real data in order to make it more applicable to deal with different applications of real-life scheduling problems.

Although the proposed algorithms achieved promising results, a more thorough evaluation is needed to fully understand their performance characteristics. As a result, future work will focus on runtime analysis to determine how computation time increases as problem size grows. Furthermore, a thorough parameter sensitivity and robustness analysis will be conducted to ensure the solution's stability and efficiency.

REFERENCES

Abdolrazzagh-Nezhad, M., & Sarbishegi, S. (2019). Job-Shop Scheduling with Fuzzy Due Date by Multi-Objective Particle Swarm Optimization. *In 2019 5th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), IEEE.* (pp. 1-5).

Abdullah, S., & Abdolrazzagh-Nezhad, M. (2014). Fuzzy job-shop scheduling problems: A review. *Information Sciences, 278*, 380-407.

Abdullah, S., Turky, A., Nazri, M. Z. A., & Sabar, N. R. (2021). An evolutionary variable neighborhood search for the unrelated parallel machine scheduling problem. *IEEE Access, 9*, 42857-42867.

Abikarram, J. B., McConky, K., & Proano, R. (2019). Energy cost minimization for unrelated parallel machine scheduling under real time and demand charge pricing. *Journal of cleaner production, 208*, 232-242.

Åblad, E., Strömberg, A. B., & Spensieri, D. (2021). Exact makespan minimization of unrelated parallel machines. *Open Journal of Mathematical Optimization, 2*(2), 1-15.

Afsar, S., Palacios, J. J., Puente, J., Vela, C. R., & González-Rodríguez, I. (2022). Multi-objective enhanced memetic algorithm for green job shop scheduling with uncertain times. *Swarm and Evolutionary Computation, 68*(101016), 1-14.

Afshari, H., Hare, W., & Tesfamariam, S. (2019). Constrained multi-objective optimization algorithms: Review and comparison with application in reinforced concrete structures. *Applied Soft Computing, 83*(2019), 1-38.

Agárdi, A., & Nehéz, K. (2021). The Unrelated Parallel Machines Scheduling Problem with Machine and Job Dependent Setup Times, Availability Constraints, Time Windows and Maintenance Times. *Management and Production Engineering Review, 12*(3).

Ahuja, R. K., Ergun, Ö., Orlin, J. B., & Punnen, A. P. (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics, 123*(1-3), 75-102.

Alhaqbani, A., Kurdi, H. A., & Hosny, M. (2022). Fish-inspired heuristics: a survey of the state-of-the-art methods. *Archives of Computational Methods in Engineering*, *29*(6), 3655-3675.

Ali, A. (2020). *Optimization of the distributed permutation flow shop scheduling problem*. University of Manitoba.

Almshhadany, S. E., & Ibrahim, L. M. (2018). Using Multi-objective Artificial Fish Swarm Algorithm to Solve the Software Project Scheduling Problem. *International Journal of Computer Applications*, 181(16), 6-13.

Alobaidi, A. T. S., & Hussein, S. A. (2017). An improved Artificial Fish Swarm Algorithm to solve flexible job shop. *In 2017 Annual Conference on New Trends in Information & Communications Technology Applications (NTICT)*, 2017, 7-12.

Al-qaness, M. A., Ewees, A. A., & Abd Elaziz, M. (2021). Modified whale optimization algorithm for solving unrelated parallel machine scheduling problems. *Soft Computing*, 25(14), 9545-9557.

Amira, M. F., Abd El-Aziz, A. A., Hesham, A. H., & Ahmed, A. F. (2018). Swarm Optimization Techniques: A survey. *Egyptian Computer Science Journal*, *42*(3), 74-91

Arık, O. A. (2022). Memetic algorithm for unrelated parallel machine scheduling problem with grey processing times. *Journal of Modelling in Management*,1-19.

Arık, O. A., & Toksarı, M. D. (2018). Multi-objective fuzzy parallel machine scheduling problems under fuzzy job deterioration and learning effects. *International Journal of Production Research*, 56(7), 2488-2505.

Arık, O. A., Schutten, M., & Topan, E. (2022). Weighted earliness/tardiness parallel machine scheduling problem with a common due date. *Expert systems with applications*, 187, 1-26.

Arnaout, J. P. (2020). A worm optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Annals of Operations Research*, 285(1-2), 273-293.

Asghari, M., & Nezhadali, S. (2016). Fuzzy Programming for Parallel Machines Scheduling: Minimizing Weighted Tardiness/Earliness and Flow time through Genetic Algorithm. *Journal of Optimization in Industrial Engineering*, 20, 19-30.

Banerjee, S., & Roy, T. K. (2012). Arithmetic operations on generalized trapezoidal fuzzy number and its applications. *Turkish Journal of Fuzzy Systems, 3*(1), 16-44.

Bansal, J. C., & Pal, N. R. (2019). Swarm and evolutionary computation. *In Evolutionary and Swarm Intelligence Algorithms, 2019*,1-10.

Beheshti, Z., & Shamsuddin, S. M. H. (2013). A review of population-based meta-heuristic algorithms. *Int. J. Adv. Soft Computing. Appl*, *5*(1), 1-35.

Behnamian, J. (2014). Particle swarm optimization-based algorithm for fuzzy parallel machine scheduling. *The International Journal of Advanced Manufacturing Technology,* 75(5), 883-895.

Behnamian, J. (2016). Survey on fuzzy shop scheduling. *Fuzzy Optimization and Decision Making*, 15(3), 331-366.

Brandimarte, P. (1992). Neighborhood search-based optimization algorithms for production scheduling: a survey. *Computer Integrated Manufacturing Systems*, 5(2), 167-176.

Cai, J., & Lei, D. (2021). A cooperated shuffled frog-leaping algorithm for distributed energy-efficient hybrid flow shop scheduling with fuzzy processing time. *Complex & Intelligent Systems*, 7(5), 2235-2253.

Cai, Y. (2010). Artificial fish school algorithm applied in a combinatorial optimization problem. *International Journal of Intelligent Systems and Applications*, *2*(1), 37.

Chakraverty, S., Sahoo, D. M., Mahato, N. R., Chakraverty, S., Sahoo, D. M., & Mahato, N. R. (2019). Concepts of Soft Computing: Defuzzification. *Fuzzy and ANN with Programming*, *pp*. 117-127.

Chaudhry, I. A., & Elbadawi, I. A. (2017). Minimization of total tardiness for identical parallel machine scheduling using genetic algorithm. *Sādhanā, 42*(1), 11-21.

Chaudhry, I. A., & Khan, A. A. (2016). A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research,* 23(3), 551-591.

Chen, C., Fathi, M., Khakifirooz, M., & Wu, K. (2022). Hybrid tabu search algorithm for unrelated parallel machine scheduling in semiconductor fabs with setup times, job release, and expired times. *Computers & Industrial Engineering*, 165, 107915.

Chen, F., Deng, P., Ding, T., & Liang, W. (2018). Research on ant colony optimization Tabu search and genetic fusion algorithm. *In 2018 2nd International Conference on Robotics and Automation Sciences (ICRAS) (pp. 1-5). IEEE.*

Chi, X., Liu, S., & Li, C. (2022). Research on optimization of unrelated parallel machine scheduling based on IG-TS algorithm. *Bulletin of the Polish Academy of Sciences. Technical Sciences,* 70(4).

Çini, G., Toy, A. Ö., & Bulut, Ö. (2022). Parallel Machine Scheduling with Fuzzy Processing Times and Sequence Dependent Setup Times: An Application in a Textile Company. *In International Conference on Intelligent and Fuzzy Systems, Springer, Cham*. (pp. 177-183).

Çolak, M., & Keskin, G. A. (2022). An extensive and systematic literature review for hybrid flow shop scheduling problems. *In International Journal of Industrial Engineering Computations* (Vol. 13, Issue 2, pp. 185–222). Growing Science. https://doi.org/10.5267/J.IJIEC.2021.12.001.

Dang, Q. V., van Diessen, T., Martagan, T., & Adan, I. (2021). A matheuristic for parallel machine scheduling with tool replacements. *European Journal of Operational Research,* 291(2), 640-660.

Deb, K. (2014). Multi-objective optimization. *In Search methodologies, Springer, Boston, MA.* (pp. 403-449).

Delahaye, D., Chaimatanan, S., & Mongeau, M. (2018). Simulated annealing: From basics to applications. In *Handbook of metaheuristics*, Cham: Springer International Publishing, (pp. 1-35).

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation, 1*(1), 3-18.

Ding, J., Shen, L., Lü, Z., Xu, L., & Benlic, U. (2019). A hybrid memetic algorithm for the parallel machine scheduling problem with job deteriorating effects. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(3), 385-397.

Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., & Cosar, A. (2019). A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, 137, 106040.

Dubois, D., & Prade, H. (1978). Operations on fuzzy numbers. *International Journal of systems science, 9*(6), 613-626.

Durasević, M., & Jakobović, D. (2018). A survey of dispatching rules for the dynamic unrelated machines environment. *Expert Systems with Applications, 113*, 555-569.

Đurasević, M., & Jakobović, D. (2021). Heuristic and Metaheuristic Methods for the Unrelated Machines Scheduling Problem: A Survey. *IEEE TRANSACTIONS ON CYBERNETICS 2021*, 1-37.

Durmaz, E. D., Aydoğan, S., & Şahin, R. (2021). An Application of Multi-Objective Scheduling with Fuzzy Measure. *International Conference on Applied Mathematics in Engineering ICAMΣ*, 21, 134.

Ekici, A., Elyasi, M., Özener, O. Ö., & Sarıkaya, M. B. (2019). An application of unrelated parallel machine scheduling with sequence-dependent setups at Vestel Electronics. *Computers & Operations Research*, 111, 130-140.

Engin, O., & Yılmaz, M. K. (2022). A fuzzy logic based methodology for multi-objective hybrid flow shop scheduling with multi-processor tasks problems and solving with an efficient genetic algorithm. *Journal of Intelligent & Fuzzy Systems, (Preprint)*, 1-13.

Erişgin Barak, M. Z., & Koyuncu, M. (2021). Fuzzy Order Acceptance and Scheduling on Identical Parallel Machines. *Symmetry*, 13(7), 1236.

Ewees, A. A., Al-qaness, M. A., & Abd Elaziz, M. (2021). Enhanced salp swarm algorithm based on firefly algorithm for unrelated parallel machine scheduling with setup times. *Applied Mathematical Modelling*, 94, 285-305.

Ezugwu, A. E., & Akutsah, F. (2018). An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times. *IEEE Access*, 6, 54459-54478.

Ezugwu, A. E., Shukla, A. K., Nath, R., Akinyelu, A. A., Agushaka, J. O., Chiroma, H., & Muhuri, P. K. (2021). Metaheuristics: a comprehensive overview and classification along with bibliometric analysis. *Artificial Intelligence Review*, 54(6), 4237-4316.

Faeq, I. F., Duaimi, M. G., &Sadiq Al-Obaidi, A. T. (2018). An Efficient Artificial Fish Swarm Algorithm with Harmony Search for Scheduling In Flexible Job-Shop Problem. *Journal of Theoretical and Applied Information Technology,96(8).*

Fanjul-Peyro, L., Ruiz, R., & Perea, F. (2019). Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. *Computers & Operations Research*, 101, 173-182.

Fister, I., Mernik, M., & Filipič, B. (2013). Graph 3-coloring with a hybrid self-adaptive evolutionary algorithm. *Computational optimization and applications*, *54*, 741-770.

Floudas, C. A., & Lin, X. (2004). Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers & Chemical Engineering*, *28*(11), 2109-2129.

Fonseca-Reyna, Y. C., Martinez, Y., Rodríguez-Sánchez, E., Méndez Hernández, B., & Coto-Palacio, L. J. (2018). An Improvement of Reinforcement Learning Approach to Permutational Flow Shop Scheduling Problem. *In Proceedings of the 13th International Conference on Operations Research (ICOR 2018), Beijing, China* (pp. 7-9).

Fu, Y., Hou, Y., Wang, Z., Wu, X., Gao, K., & Wang, L. (2021). Distributed scheduling problems in intelligent manufacturing systems. *Tsinghua Science and Technology*, 26(5), 625-645.

Gani, A. N., & Assarudeen, S. M. (2012). A new operation on triangular fuzzy number for solving fuzzy linear programming problem. *Applied Mathematical Sciences*, 6(11), 525-532.

Gao, D., Wang, G. G., & Pedrycz, W. (2020). Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism. *IEEE Transactions on Fuzzy Systems*, 28(12), 3265-3275. https://doi.org/10.1109/TFUZZ.2020.3003506.

Gao, Y., Xie, L., Zhang, Z., & Fan, Q. (2020). Twin support vector machine based on improved artificial fish swarm algorithm with application to flame recognition. *Applied Intelligence*, *50*, 2312-2327.

Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. *Freeman, San Francisco*, 1979.

Ge, H., Sun, L., Chen, X., & Liang, Y. (2016). An efficient artificial fish swarm model with estimation of distribution for flexible job shop scheduling. *International Journal of Computational Intelligence Systems,* 9(5), 917-931.

Geurtsen, M., Adan, J., & Akçay, A. (2023). Integrated maintenance and production scheduling for unrelated parallel machines with setup times. *Flexible Services and Manufacturing Journal*, 1-34.

Geyik, F., & Elibal, K. (2017). A linguistic approach to non-identical parallel processor scheduling with fuzzy processing times. *Applied Soft Computing*, 55, 63-71.

Ghaffar, A., Sattar, M. U., Munir, M., & Qureshi, Z. (2022). Multi-objective fuzzy-based adaptive memetic algorithm with hyper-heuristics to solve university course timetabling problem. *EAI Endorsed Transactions on Scalable Information Systems*, 9(4). https://doi.org/10.4108/eai.16-12-2021.172435.

Gharehgozli, A. H., Tavakkoli-Moghaddam, R., & Zaerpour, N. (2009). A fuzzy-mixed-integer goal programming model for a parallel-machine scheduling problem with sequence-dependent setup times and release dates. *Robotics and Computer-Integrated Manufacturing*, 25(4-5), 853-859.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. Computers & operations research, 13(5), 533-549.

Golneshini, F. P., & Fazlollahtabar, H. (2019). Meta-heuristic algorithms for a clustering-based fuzzy bi-criteria hybrid flow shop scheduling problem. *Soft Computing*, 23(22), 12103-12122.

Gomes, F. R. A. (2017). Scheduling problem with unrelated parallel machines: mathematical formulation. decomposition methods and hybridization. Universidade Federal de Minas Gerais.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *In Annals of discrete mathematics. Elsevier*. (Vol. 5, pp. 287-326).

Guevara-Guevara, A. F., Gómez-Fuentes, V., Posos-Rodríguez, L. J., Remolina-Gómez, N., & González-Neira, E. M. (2022). Earliness/tardiness minimization in a no-wait flow shop with sequence-dependent setup times. *Journal of Project Management*, 7(3), 177–190. https://doi.org/10.5267/j.jpm.2021.12.001.

Guo, J., Zhong, J., Li, Y., Du, B., & Guo, S. (2018). A hybrid artificial fish swam algorithm for disassembly sequence planning considering setup time. *Assembly Automation*, 39(1), 140-153.

Hajisalem, V., & Babaie, S. (2018). A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection. *Computer Networks*, 136, 37-50.

Hasan, M. M., & Arefin, M. R. (2017). Application of linear programming in scheduling problem. Dhaka University Journal of Science, 65(2), 145-150.

Heydari, M., & Aazami, A. (2018). Minimizing the maximum tardiness and makespan criteria in a job shop scheduling problem with sequence dependent setup times. *Journal of industrial and systems engineering*, 11(2), 134-150.

Holland, J. H. (1970). Adaptation in natural and artificial systems. University of Michigan Press google schola, 2, 29-41.

Huang, J., Zeng, J., Bai, Y., Cheng, Z., Feng, Z., Qi, L., & Liang, D. (2021). Layout optimization of fiber Bragg grating strain sensor network based on modified artificial fish swarm algorithm. *Optical Fiber Technology*, *65*, 102583.

Huang, Z., & Chen, Y. (2013). An improved artificial fish swarm algorithm based on hybrid behavior selection. *International Journal of Control and Automation*, *6*(5), 103-116.

Hussain, K., Mohd Salleh, M. N., Cheng, S., & Shi, Y. (2019). Metaheuristic research: a comprehensive survey. *Artificial intelligence review*, 52(4), 2191-2233.

Iranmanesh, S. H., Tanhaie, F., & Rabbani, M. (2017). Improving Artificial Fish Swarm Algorithm by Applying Group Escape Behavior of Fish. In *Recent Advances in Soft Computing: Proceedings of the 22nd International Conference on Soft Computing, Czech Republic, Springer International Publishing, 22* (pp. 46-54).

Jadhava, V. S., Buktareb, S. U., & Fegade, M. R. (2024). Multi-Objective Single Machine Scheduling Problem under Fuzzy Programming Technique. *International Journal of Mathematics, Statistics and Operations Research*. 4(1), 67-76.

Jayanthi, S. E., & Karthigeyan, S. (2017). Minimizing Weighted Earliness and Tardiness under Fuzziness using Firefly Algorithm. *International Journal of Applied Engineering Research*, 12(24), 13974-13980.

Jia, Z., Yan, J., Leung, J. Y. T., Li, K., & Chen, H. (2019). Ant colony optimization algorithm for scheduling jobs with fuzzy processing time on parallel batch machines with different capacities. *Applied Soft Computing Journal*,75, 548–561. https://doi.org/10.1016/j.asoc.2018.11.027.

Jiang, M., & Cheng, Y. (2010). Simulated annealing artificial fish swarm algorithm. *In 2010 8th World Congress on Intelligent Control and Automation. IEEE*. pp. 1590-1593.

Jin, J., Zhang, Z., & Zhang, L. (2023, September). A modified artificial fish swarm algorithm for unit commitment optimization. In *Eighth International Conference on Electromechanical Control Technology and Transportation (ICECTT 2023)* (Vol. 12790, pp. 760-767). SPIE.

Jouhari, H., Lei, D., Al-qaness, M. A., Elaziz, M. A., Damaševičius, R., Korytkowski, M., & Ewees, A. A. (2020). Modified Harris Hawks optimizer for solving machine scheduling problems. *Symmetry*, *12*(9), 1460.

Kanakavalli, P., Vommi, V., & Medikondu, N. (2018). Fuzzy heuristic algorithm for simultaneous scheduling problems in flexible manufacturing system. *Management Science Letters*, 8(12), 1319-1330.

Karacan, I., Senvar, O., & Bulkan, S. (2023). A novel parallel simulated annealing methodology to solve the no-wait flow shop scheduling problem with earliness and tardiness objectives. *Processes*, *11*(2), 454.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *In Proceedings of ICNN'95-international conference on neural networks* (Vol. 4, pp. 1942-1948). *IEEE*.

Khalifa, H. (2020). On single machine scheduling problem with distinct due dates under fuzzy environment. *International Journal of Supply and Operations Management*, 7(3), 272-278.

Kılıç, G., & Organ, A. (2025). A new neighborhood-based VNS algorithm for unrelated parallel machine scheduling problem with sequence-dependent setup times. *Pamukkale University Journal of Engineering Sciences*, 1000(1000), 0-0.

Kongsri, P., & Buddhakulsomsiri, J. (2020). A mixed integer programming model for unrelated parallel machine scheduling problem with sequence dependent setup time to minimize makespan and total tardiness. *In 2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA). IEEE.* (pp. 605-609).

Krause, J., Cordeiro, J., Parpinelli, R. S., & Lopes, H. S. (2013). A survey of swarm algorithms applied to discrete optimization problems. *In Swarm intelligence and bio-inspired computation. Elsevier*. (pp. 169-191).

Krishnaveni, H., & Janita, V. S. (2019). Modified Artificial Fish Swarm Algorithm for Efficient Task Scheduling in Cloud Environment. *International Journal of Computer Sciences and Engineering*, *7*(5), 1363-71.

Kumar, H., Kumar, P., & Sharma, M. (2018). An effective new hybrid optimization algorithm for solving flow shop scheduling problems. *Malaya Journal of Matematik*, S (1), 107-113.

Kumar, M., Husain, D. M., Upreti, N., & Gupta, D. (2010). Genetic algorithm: Review and application. *Available at SSRN 3529843*.

Kumar, V., & Kumar, D. (2021). A systematic review on firefly algorithm: past, present, and future. *Archives of Computational Methods in Engineering, 28*, 3269-3291.

Laguna, M. (2018). Tabu search. In Handbook of heuristics. Springer, Cham, (pp. 741-758). https://doi.org/10.1007/978-3-319-07124-4_24.

Lee, H. T., Chen, S. H., & Kang, H. Y. (2002). Multicriteria scheduling using fuzzy theory and tabu search. *International Journal of Production Research,40*(5), 1221–1234.

Lei, D., & Guo, X. (2012). Swarm-based neighbourhood search algorithm for fuzzy flexible job shop scheduling. *International Journal of Production Research, 50*(6), 1639-1649.

Lei, D., Yuan, Y., & Cai, J. (2021). An improved artificial bee colony for multi-objective distributed unrelated parallel machine scheduling. *International Journal of Production Research, 59*(17), 5259-5271.

Li, K., Chen, J., Fu, H., Jia, Z., & Fu, W. (2019). Uniform parallel machine scheduling with fuzzy processing times under resource consumption constraint. *Applied Soft Computing, 82*, 1-13.

Li, K., Chen, J., Fu, H., Jia, Z., & Wu, J. (2020). Parallel machine scheduling with position-based deterioration and learning effects in an uncertain manufacturing system. *Computers and Industrial Engineering*, 149. https:/doi.org/10.1016/j.cie.2020.106858

Li, R., Gong, W., & Lu, C. (2022). Self-adaptive multi-objective evolutionary algorithm for flexible job shop scheduling with fuzzy processing time. *Computers & Industrial Engineering*, 168, 108099.

Li, T., Yang, F., Zhang, D., & Zhai, L. (2021). Computation scheduling of multi-access edge networks based on the artificial fish swarm algorithm. *IEEE Access*, *9*, 74674-74683.

Li Xiao Lei, (2002). A new intelligent optimization-artificial fish school algorithm. Doctor thesis, Zhejiang University, Hangzhou, China.

Li, X., & Yin, M. (2013). A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem. *International Journal of Production Research*, 51(16), 4732-4754.

Li, X., Wu, Y., Wei, M., Guo, Y., Yu, Z., Wang, H., ... & Fan, H. (2021). A novel index of functional connectivity: phase lag based on Wilcoxon signed rank test. *Cognitive Neurodynamics*, *15*, 621-636.

Li, Z., & Ierapetritou, M. (2008). Process scheduling under uncertainty: Review and challenges. *Computers & Chemical Engineering*, *32*(4-5), 715-727.

Liao, T. W., & Su, P. (2017). Parallel machine scheduling in fuzzy environment with hybrid ant colony optimization including a comparison of fuzzy number ranking methods in consideration of spread of fuzziness. *Applied Soft Computing Journal, 56*, 65–81.

Lin, D. Y., & Huang, T. Y. (2021). A hybrid metaheuristic for the unrelated parallel machine scheduling problem. *Mathematics*, *9*(7), 1-20.

Lin, L. & Gen, M. (2018). Hybrid evolutionary optimisation with learning for production scheduling: state-of-the-art survey on algorithms and applications. *International Journal of Production Research, 56*(1–2), 193–223.

Lin, Y. K., & Lin, H. C. (2015). Bicriteria scheduling problem for unrelated parallel machines with release dates. *Computers & Operations Research, 64*, 28-39.

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.

Liou, T. S., & Wang, M. J. J. (1992). Ranking fuzzy numbers with integral value. *Fuzzy sets and systems*, *50*(3), 247-255.

Liu, L., Chang, Z., & Song, S. (2022). Optimization of a molten iron scheduling problem with uncertain processing time using variable neighborhood search algorithm. *Scientific reports*, 12(1), 7303.

Liu, Y., Tao, Z., Yang, J., & Mao, F. (2019). The modified artificial fish swarm algorithm for least-cost planning of a regional water supply network problem. *Sustainability*, *11*(15), 4121.

Liu, Z., Liang, X., Hou, L., Yang, D., & Tong, Q. (2023). Multi-strategy dynamic evolution-based improved MOEA/D algorithm for solving multi-objective fuzzy flexible job shop scheduling problem. *IEEE Access*, 11, 54596-54606.

Ma, W., Liu, Y., & Zhang, X. (2019). A new model and algorithm for uncertain random parallel machine scheduling problem. *Soft Computing, 23*(15), 6555-6566.

Malhotra, K., Gupta, D., Goel, S., & Tripathi, A. K. (2022). Bi-Objective Flow Shop Scheduling with Equipotential Parallel Machines. *Malaysian Journal of Mathematical Sciences, 16*(3), 451–470.

Manupati, V. K., Rajyalakshmi, G., Chan, F. T., & Thakkar, J. J. (2017). A hybrid multi-objective evolutionary algorithm approach for handling sequence-and machine-dependent set-up times in unrelated parallel machine scheduling problem. *Indian Academy of Sciences, 42*(3), 391-403.

Masrom, S., Moser, I., Montgomery, J., Abidin, S. Z. Z., & Omar, N. (2013, December). Hybridization of particle swarm optimization with adaptive genetic algorithm operators. In *2013 13th International Conference on Intellient Systems Design and Applications* (pp. 153-158). IEEE.

Meng, R., Rao, Y., & Luo, Q. (2020). Modeling and solving for bi-objective cutting parallel machine scheduling problem. *Annals of Operations Research, 285*(1), 223-245.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6), 1087-1092.

Molaee, E., & Moslehi, G. (2014). Minimizing the number of tardy jobs on a single machine with an availability constraint. *Journal of Industrial Engineering, 2014*, 1-14.

Mortada, S., & Yusof, Y. (2021). A Neighborhood Search for Artificial Bee Colony in Vehicle Routing Problem with Time Windows. *International Journal of Intelligent Engineering & Systems*, 14(3).

Munoz-Estrada, L. F. (2020). Multi-objective Resource Constrained Parallel Machine Scheduling Model with Setups, Machine Eligibility Restrictions, Release and Due Dates with User Interaction. Doctoral dissertation, Arizona State University.

Naderi-Beni, M., Ghobadian, E., Ebrahimnejad, S., & Tavakkoli-Moghaddam, R. (2014). Fuzzy bi-objective formulation for a parallel machine scheduling problem with machine eligibility restrictions and sequence-dependent setup times. *International Journal of Production Research*, *52*(19), 5799-5822.

Nailwal, K. K., Gupta, D., & Sharma, S. (2015). Fuzzy bi-criteria scheduling on parallel machines involving weighted flow time and maximum tardiness. *Cogent Mathematics*, *2*(1), 1019792.

Najafi, H. S., Edalatpanah, S. A., & Dutta, H. (2016). A nonlinear model for fully fuzzy linear programming with fully unrestricted variables and parameters. *Alexandria engineering journal*, 55(3), 2589-2595.

Najari, F., Alaghebandha, M., Mohammadi, M., & Sobhanallahi, M. A. (2018). Fuzzy Multi-objective Permutation Flow Shop Scheduling Problem with Fuzzy Processing Times under Learning and Aging Effects. *Journal of Quality Engineering and Production Optimization*, 3(2), 77-98.

Nakane, T., Bold, N., Sun, H., Lu, X., Akashi, T., & Zhang, C. (2020). Application of evolutionary and swarm optimization in computer vision: a literature survey. *IPSJ Transactions on Computer Vision and Applications*, 12(1), 1-34.

Nasseri, S. H., & Ebrahimnejad, A. (2011). Sensitivity analysis on linear programming problems with trapezoidal fuzzy variables. *International Journal of Operations Research and Information Systems* (*IJORIS*), 2(2), 22-39.

Ning, X., Desrosiers, C., & Karypis, G. (2015). A comprehensive survey of neighborhood-based recommendation methods. *Recommender systems handbook*, 37-76.

Noor, N. M., Alwadood, Z., Fuazee, D. D. M., & Sayuti, N. A. M. (2022). A Modified Work Schedule for Police Officer at a Local University using Mathematical Programming Model. *International Journal of Business and Technology Management*, 4(2), 1-6.

Ojstersek, R., Brezocnik, M., & Buchmeister, B. (2020). Multi-objective optimization of production scheduling with evolutionary computation: A review. *International Journal of Industrial Engineering Computations*, 11(3), 359-376.

Pavlov, A. A., Misura, E. B., Melnikov, O. V., Mukha, I. P., & Lishchuk, K. I. (2020). Approximation algorithm for parallel machines total tardiness minimization problem for planning processes automation. *In Advances in Computer Science for Engineering and Education II*. *Springer International Publishing*, (pp. 459-467).

Peddi, P. B. R. (2019). Evaluating fuzzy risk based on a new method of ranking fuzzy numbers using centroid of centroids. *International Journal of Mathematics in Operational Research*, *14*(4), 451-472.

Pei, Z., Wan, M., & Wang, Z. (2020). A new approximation algorithm for unrelated parallel machine scheduling with release dates. *Annals of Operations Research, 285*(1), 397-425.

Peng, B., Lü, Z., & Cheng, T. C. E. (2015). A tabu search/path relinking algorithm to solve the job shop scheduling problem. *Computers & Operations Research*, *53*, 154-164.

Peng, Z., Dong, K., Yin, H., & Bai, Y. (2018). Modification of fish swarm algorithm based on levy flight and firefly behavior. *Computational Intelligence and Neuroscience*, *2018*(1), 9827372.

Pınarbaşı, A. (2019). Assignment of Patients to Hospitals in a Case of a Large-Scale Earthquake. Master's thesis, Eastern Mediterranean University (EMU)-Doğu Akdeniz Üniversitesi (DAÜ).

Pinedo, M. L. (2008). Scheduling theory, algorithms, and systems (3rd ed.). *New York: Springer*.

Pirim, H., Eksioglu, B., & Bayraktar, E. (2008). Tabu Search: a comparative study.

Pourpanah, F., Wang, R., Lim, C. P., Wang, X. Z., & Yazdani, D. (2023). A review of artificial fish swarm algorithms: recent advances and applications. *Artificial Intelligence Review, 2022*, 1-37.

Prajapati, V. K., Jain, M., & Chouhan, L. (2020). Tabu search algorithm (TSA): A comprehensive survey. In *2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE)*, IEEE. (pp. 1-8).

Pulansari, F. (2021). The Unrelated Parallel Machine Scheduling with a Dependent Time Setup using Ant Colony Optimization Algorithm. *Jurnal Teknik Industri, 23*(1), 65-74.

Pulansari, F., & Triyono, M. D. R. (2021). Particle Swarm Optimization to Solve Unrelated Parallel Machine Scheduling Problems. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1125, No. 1, p. 012109). *IOP Publishing*.

Pythaloka, D., Wibowo, A. T., & Sulistiyo, M. D. (2015). Artificial fish swarm algorithm for job shop scheduling problem. *In 2015 3rd International Conference on Information and Communication Technology, 2015*, 437-443.

Rajabi Moshtaghi, H., Toloie Eshlaghy, A., & Motadel, M. R. (2021). A comprehensive review on meta-heuristic algorithms and their classification with novel approach. *Journal of Applied Research on Industrial Engineering, 8*(1), 63-89.

Rajba, P. (2021). Sampling Method for the Robust Single Machine Scheduling with Uncertain Parameters. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12747 LNCS, 594–607. https://doi.org/10.1007/978-3-030-77980-1_45.

Rajba, P. (2021). Uncertainty Modeling in Single Machine Scheduling Problems. A Survey. *In International Conference on Dependability and Complex Systems. Springer, Cham* (pp. 359-368),331-366 .

Ranjith, K., & Karthikeyan, K. (2024). Minimizing the total waiting time for fuzzy two-machine flow shop scheduling problem with uncertain processing time. *Baghdad Science Journal*.

Rezaeian, J., Mohammad-Hosseini, S., Zabihzadeh, S., & Shokoufi, K. (2020). Fuzzy scheduling problem on unrelated parallel machine in JIT production system. *Artificial Intelligence Evolution, 1*(1), 17-33.

Rifai, A. P., Kusumastuti, P. A., Mara, S. T. W., Norcahyo, R., & Dawal, S. Z. M. (2021). Multi-operator hybrid genetic algorithm-simulated annealing for reentrant permutation flow-shop scheduling. *ASEAN Engineering Journal*, *11*(3), 109-126.

Rostami, M., Pilerood, A. E., & Mazdeh, M. M. (2015). Multi-objective parallel machine scheduling problem with job deterioration and learning effect under fuzzy environment. *Computers and Industrial Engineering*, *85*, 206–215.

Sabti, A. N. (2017). Solution approaches for multi objective parallel machine scheduling problems. Doctoral dissertation, Anadolu University.

Sadati, A., Tavakkoli-Moghaddam, R., Naderi, B., & Mohammadi, M. (2019). A bi-objective model for a scheduling problem of unrelated parallel batch processing machines with fuzzy parameters by two fuzzy multi-objective meta-heuristics. *Iranian Journal of Fuzzy Systems*, *16*(4), 21-40.

Safarzadeh, H., & Niaki, S. (2023). Unrelated parallel machine scheduling with machine processing cost. *International Journal of Industrial Engineering Computations*, *14*(1), 33-48.

Salimifard, K., Mohammadi, D., Moghdani, R., & Abbasizad, A. (2019). Green fuzzy parallel machine scheduling with sequence-dependent setup in the plastic moulding industry. *Asian Journal of Management Science and Applications*, *4*(1), 27-48.

Serna, N. J. E., Seck-Tuoh-Mora, J. C., Medina-Marin, J., Hernandez-Romero, N., Barragan-Vite, I., & Armenta, J. R. C. (2021). A global-local neighborhood search algorithm and tabu search for flexible job shop scheduling problem. *PeerJ Computer Science*, *7*, e574.

Shafipour, M., Rezaeian, J., & Foroutan, R. A. (2022). Minimizing JIT scheduling of unrelated parallel machine with family setups and soft time window constraints. *Research Square*, *2022*, 1-33.

Shami, T. M., El-Saleh, A. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M. A., & Mirjalili, S. (2022). Particle swarm optimization: A comprehensive survey. *IEEE Access*, *10*, 10031-10061.

Shao, Z., Shao, W., & Pi, D. (2020). Effective heuristics and metaheuristics for the distributed fuzzy blocking flow-shop scheduling problem. *Swarm and Evolutionary Computation, 59*, 1-17.

Sharif, H. O., Ghareb, M. I., & Mohamedyusf, H. M. (2024). A Hybrid Artificial Bee Colony and Artificial Fish Swarm Algorithms for Software Cost Estimation. *UHD Journal of Science and Technology,* 8(1), 129–141. doi.org/10.21928/uhdjst.v8n1y2024.pp129-141

Shen, J. (2019). An uncertain parallel machine problem with deterioration and learning effect. *Computational and Applied Mathematics*, 38, 1-17.

Singh, N., Son, L. H., Chiclana, F., & Magnot, J. P. (2020). A new fusion of salp swarm with sine cosine for optimization of non-linear functions. *Engineering with Computers, 36*(1), 185-212.

Soto, C., Dorronsoro, B., Fraire, H., Cruz-Reyes, L., Gomez-Santillan, C., & Rangel, N. (2020). Solving the multi-objective flexible job shop scheduling problem with a novel parallel branch and bound algorithm. *Swarm and Evolutionary Computation, 53*, 1-32.

Srivastava, G., & Singh, A. (2023). An evolutionary approach comprising tailor-made variation operators for rescue unit allocation and scheduling with fuzzy processing times. *Engineering Applications of Artificial Intelligence*, 123, 106246.

Sun, Y., Qian, X., & Liu, S. (2018). Scheduling deteriorating jobs and module changes with incompatible job families on parallel machines using a hybrid SADE-AFSA algorithm. *In International Conference on Learning and Intelligent Optimization, Springer, Cham. (pp. 455-472).*

Sun, Y., Qian, X., & Liu, S. (2019). Scheduling Deteriorating Jobs and Module Changes with Incompatible Job Families on Parallel Machines Using a Hybrid SADE-AFSA Algorithm. In *Learning and Intelligent Optimization: 12th International Conference, LION 12, Kalamata, Greece, June 10–15, 2018, Revised Selected Papers 12* (pp. 455-472). Springer International Publishing.

Sureja, N. M., & Patel, S. P. (2020). Solving a combinatorial optimization problem using artificial fish swarm algorithm. *International Journal of Engineering Trends and Technology*, *68*(5), 27-32.

Tahami, H., & Fakhravar, H. (2022). A Literature review on combining heuristics and exact algorithms in combinatorial optimization. *European Journal of Information Technologies and Computer Science, 2*(2), 6-12.

Talbi, E. G. (2015). Hybrid metaheuristics for multi-objective optimization. *Journal of Algorithms & Computational Technology*, *9*(1), 41-63.

Tan, W. H., & Mohamad-Saleh, J. (2020). Normative fish swarm algorithm (NFSA) for optimization. *Soft Computing*, *24*(3), 2083-2099.

Tang, J., Liu, G., & Pan, Q. (2021). A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. *IEEE/CAA Journal of Automatica Sinica, 8*(10), 1627-1643.

Tein, L.H, (2015). Enhanced Evolutionary Algorithm With Cuckoo Search For Nurse Scheduling And Rescheduling Problem. Doctor of Philosophy, University Utara Malaysia.

Tirkolaee, E. B., Goli, A., & Weber, G. W. (2020). Fuzzy mathematical programming and self-adaptive artificial fish swarm algorithm for just-in-time energy-aware flow shop scheduling problem with outsourcing option. *IEEE transactions on fuzzy systems, 28*(11), 2772-2783.

Tom, M. (2019). Fuzzy Multi-Constraint Programming Model for Weekly Meals Scheduling (2019). In 2019 International Conference on Fuzzy Systems, 2019, 1-6.

Torabi, S. A., Sahebjamnia, N., Mansouri, S. A., & Bajestani, M. A. (2013). A particle swarm optimization for a fuzzy multi-objective unrelated parallel machine scheduling problem. *Applied Soft Computing, 13*(12), 4750-4762.

Ulaga, L., Đurasević, M., & Jakobović, D. (2022). Local search based methods for scheduling in the unrelated parallel machines environment. *Expert Systems with Applications, 199*, 1-15.

Unlu, Y., & Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers & Industrial Engineering*, *58*(4), 785-800.

Varol, B., & Subulan, K. (2021). Multi-Objective Fuzzy-Stochastic Goal Programming Model For Flexible Job Shop Scheduling With Sequence-Dependent Setup Times. *5th International Conference on Computational Mathematics and Engineering.*

Vela, C. R., Afsar, S., Palacios, J. J., Gonzalez-Rodriguez, I., & Puente, J. (2020). Evolutionary tabu search for flexible due-date satisfaction in fuzzy job shop scheduling. *Computers & Operations Research*, *119*, 104931.

Villablanca, s. P. M. (2016). *An artificial fish swarm algorithm to solve the set covering problem*. doctoral dissertation, pontificia universidad católica de valparaíso.

Vitaliiovich, H. S., & Leonidovna, S. O. (2020). Hybrid algorithm based on fish school search and grey wolf optimizer algorithms for food enterprise management. *European science review*, *7*(8), 19-22.

Wan, Z., Wang, G., & Sun, B. (2013). A hybrid intelligent algorithm by combining particle swarm optimization with chaos searching technique for solving nonlinear bilevel programming problems. *Swarm and Evolutionary Computation*, *8*, 26-32.

Wang, F., Zhao, L., & Bai, Y. (2022). Path planning for unmanned surface vehicles based on modified artificial fish swarm algorithm with local optimizer. *Mathematical Problems in Engineering*, *2022*(1), 1283374.

Wang, R., Jia, Z., & Li, K. (2021). Scheduling parallel-batching processing machines problem with learning and deterioration effect in fuzzy environment. *Journal of Intelligent and Fuzzy Systems*,*40*(6), 12111–12124.

Wang, S. G., Jiang, Y. C., Wang, F. J., & Jiang, S. F. (2016). ACO-AFSA Algorithm in Function Optimization and Its Application. In *MATEC Web of Conferences*, *EDP Sciences*, (Vol. 63, p. 05014).

Wu, R., Tian, Z., Li, X., Wu, C., Tang, H., & Li, Y. (2025). Improved discrete particle swarm optimization algorithm for solving fuzzy flexible job shop machines and automated

guided vehicles fusion scheduling problem. *Engineering Applications of Artificial Intelligence*, 160, 111951.

Xiaoyue, F. U. (2020). Optimization of the production scheduling problem with maintenance.

Xu, G., Guan, Z., Yue, L., Mumtaz, J., & Liang, J. (2021). Modeling and Optimization for Multi-Objective Nonidentical Parallel Machining Line Scheduling with a Jumping Process Operation Constraint. *Symmetry*, *13*(8) 1-24

Xu, W., Ji, Z., & Wang, Y. (2018). A flower pollination algorithm for flexible job shop scheduling with fuzzy processing time. *Modern Physics Letters*, 32(34n36), 1840113.

Yadollahi, M., & Razavi, S. S. (2019). Using Artificial Fish Swarm Algorithm to Solve University Exam Timetabling Problem. *Journal of Advances in Computer Research, 10*(2), 109-117.

Yaghtin, M., & Javid, Y. (2023). Genetic Algorithm based on Greedy Strategy in Unrelated Parallel-Machine Scheduling Problem Using a Fuzzy Approach with Periodic Maintenance and Process Constraints. *International Journal of Supply & Operations Management*, 10(3).

Yaghtin, M., & Javid, Y. (2025). Multiobjective unrelated parallel machines scheduling problem with periodic maintenance activities and dependent processing times. *Journal of Modelling in Management*, 20(2), 477-494.

Yang, X. S. (2008). Nature-Inspired Metaheuristic Algorithms, *Luniver Press*, UK.

Yang, X. S., Deb, S., Fong, S., He, X., & Zhao, Y. X. (2016). From swarm intelligence to metaheuristics: nature-inspired optimization algorithms. *Computer*, *49*(9), 52-59.

Yeh, W. C., Lai, P. J., Lee, W. C., & Chuang, M. C. (2014). Parallel-machine scheduling to minimize makespan with fuzzy processing times and learning effects. *Information Sciences, 269*, 142–158.

Yepes-Borrero, J. C., Villa, F., Perea, F., & Caballero-Villalobos, J. P. (2020). GRASP algorithm for the unrelated parallel machine scheduling problem with setup times and additional resources. *Expert Systems with Applications, 141*, 1-12.

Zadeh, L. A. (1965). Fuzzy sets. *Information and control*, *8*(3), 338-353.

Zadeh, L. A. (1976). A fuzzy-algorithmic approach to the definition of complex or imprecise concepts. *International Journal of Man-machine studies*, *8*(3), 249-291.

Zhao, L., Wang, F., & Bai, Y. (2023). Route planning for autonomous vessels based on improved artificial fish swarm algorithm. *Ships and Offshore Structures*, *18*(6), 897-906.

Zheng, F., Jin, K., Xu, Y., & Liu, M. (2022). Unrelated parallel machine scheduling with processing cost, machine eligibility and order splitting. *Computers & Industrial Engineering*, *171*, 108483.

Zhu, K., & Jiang, M. (2010). The optimization of job shop scheduling problem based on Artificial Fish Swarm Algorithm with tabu search strategy. *In Third International Workshop on Advanced Computational Intelligence, 2010*, 323-327.

Zimmermann, H. J. (2010). Fuzzy set theory. *Wiley interdisciplinary reviews: computational statistics*, *2*(3), 317-332.

Zuo, J., Chen, J., Tan, Y., Wang, M., & Wen, L. (2019). A multi-agent collaborative work planning strategy based on AFSA-PSO algorithm. *In 2019 International conference on Robots & Intelligent System, 2019*, 254-257.

Appendix A

An example for a multi-objective UPMSP using triangular fuzzy numbers to illustrate fuzzy calculations

Step 1: Set up the problem. Assume we have two unrelated machines (M_1, M_2) and three jobs (J_1, J_2, J_3) with assignments $J_1 \rightarrow M_1$ and $J_2, J_3 \rightarrow M_2$

Step 2: Fuzzy parameters as a Triangular Fuzzy Numbers (TFNs) for

- Processing Times ($\tilde{p}_{ij}$)

  $\tilde{p}_{11} = (4,5,6), \tilde{p}_{22} = (6,8,10), \ \tilde{p}_{32} = (3,4,6)$

- due dates ($\tilde{d}_i$)

  $\tilde{d}_1 = (7,8,9), \tilde{d}_2 = (10,12,14), \ \tilde{d}_3 = (8,9,10)$

Step 3: compute fuzzy makespan and total tardiness

- M_1 : fuzzy completion time $\tilde{C}_{11}$ =(4,5,6)
- M_2 : $\tilde{C}_{22}$ =(6,8,10), $\tilde{C}_{32} = \tilde{C}_{22} + \tilde{p}_{32} = (6,8,10) + (3,4,6)$

$$= (6+3, 8+4, 10+6) = (9,12,16)$$

$$\tilde{C}_{max} = max\left(\tilde{C}_{11}, \tilde{C}_{32}\right) = (9,12,16)$$

- Fuzzy tardiness $\tilde{T}_{ij} = \max\left(0, \tilde{C}_{ij} - \tilde{d}_i\right)$

$$\tilde{T}_{11} = (0,4-7,5-8,6-9) = 0$$

$$\tilde{T}_{22} = (0,6-10,8-12,10-14) = 0$$

$$\tilde{T}_{32} = (0,9-8,12-9,16-10) = (1,3,6)$$

Total tardiness $\left(\sum_{i=1}^{2} \tilde{T}_{ij}\right) = (1,3,6)$

So: the multi-objective

$$\tilde{F} = w\tilde{C}_{max} + (1 - \mathrm{w}) \sum_{i=1}^{2} \tilde{T}_{ij}$$

If w=0.6

$$\tilde{F} = 0.6(9,12,16) + 0.4(1,3,6)$$

$$= (5.4,7.2,9.6) + (0.4,1.2,2.4) = (5.8,8.4,12)$$

Step 4: A defuzzification method (the total integral value method) is used to convert the fuzzy multi-objective makespan and total tardiness into crisp value by formula:

$$I_T^{\alpha}(A) = \tfrac{1}{2}\alpha(b + c) + \tfrac{1}{2}(1 - \alpha)(a + b),$$

$$= \frac{1}{2}[\alpha c + b + (1 - \alpha)a],$$

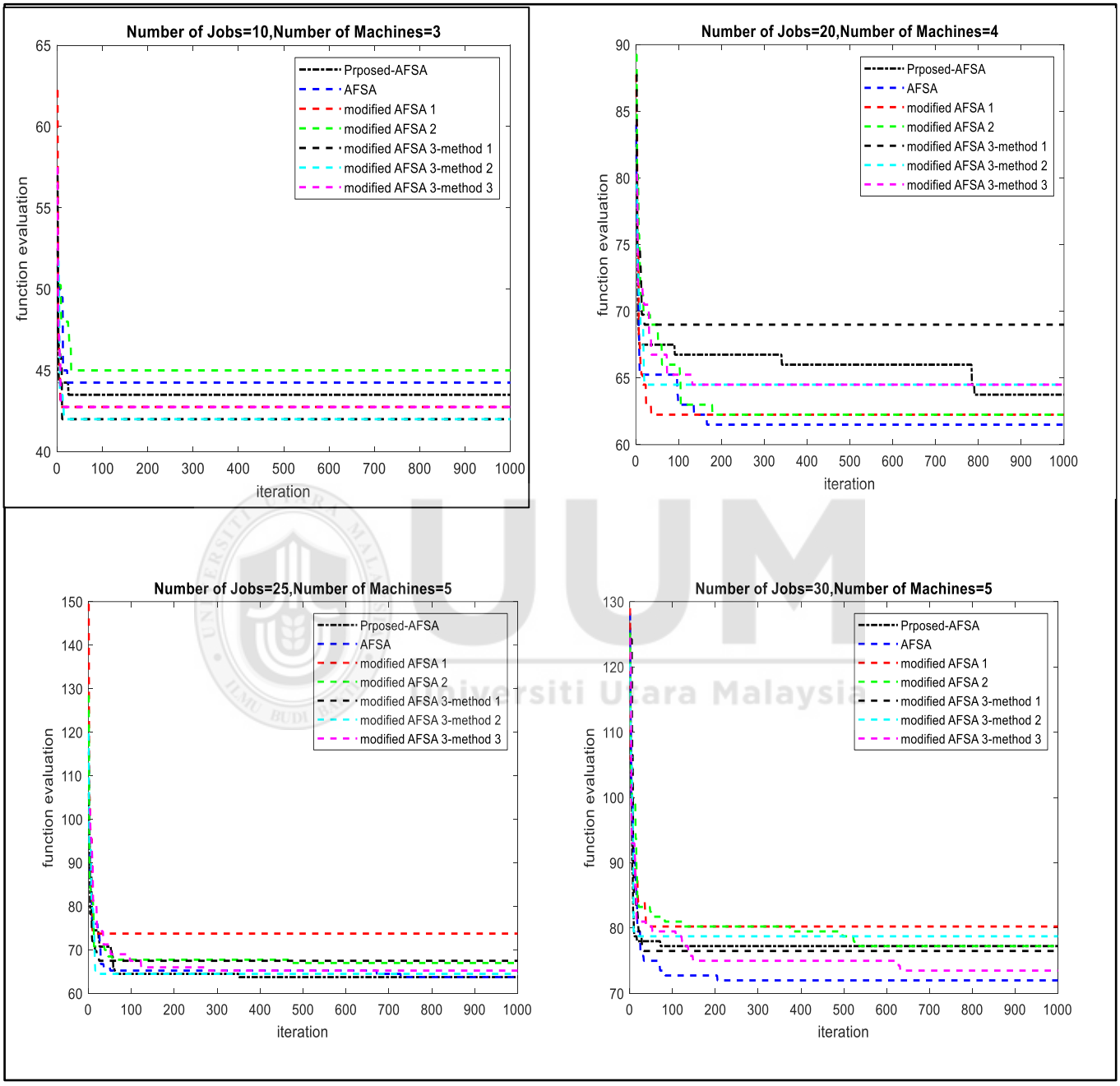
and $\alpha \in 0.5$

$$I_T^{0.5}(5.8,8.4,12) = \frac{1}{2}[0.5(12) + 8.4 + (1 - 0.5)(5.8)]$$

$$= \frac{1}{2}[6 + 8.4 + 2.9] = \frac{17.3}{2} = 8.65$$

Appendix B

The remaining performance figures of the proposed algorithms for the nine tests



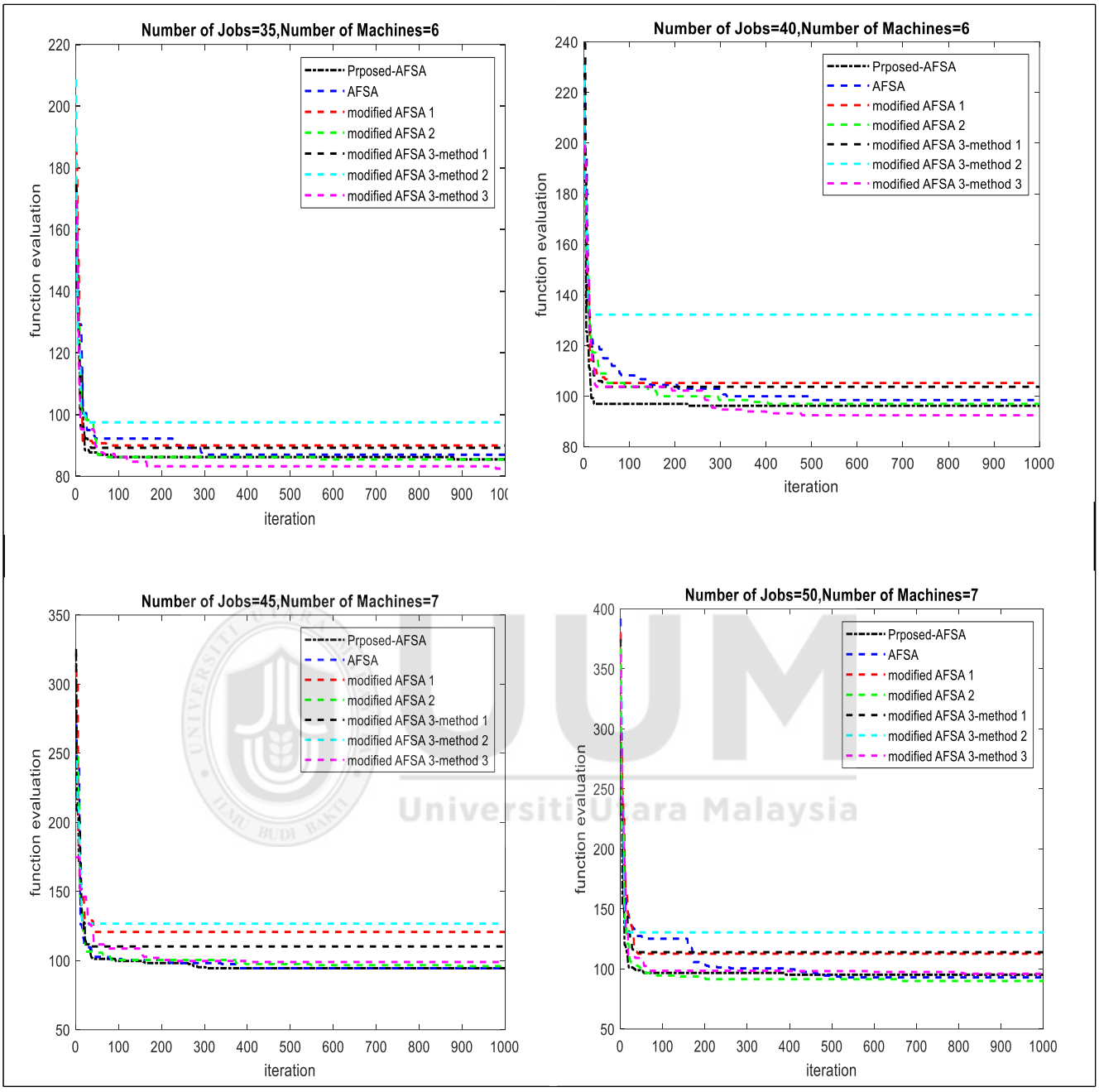


Figure B1: The performance of the proposed modified algorithm for small size-problem

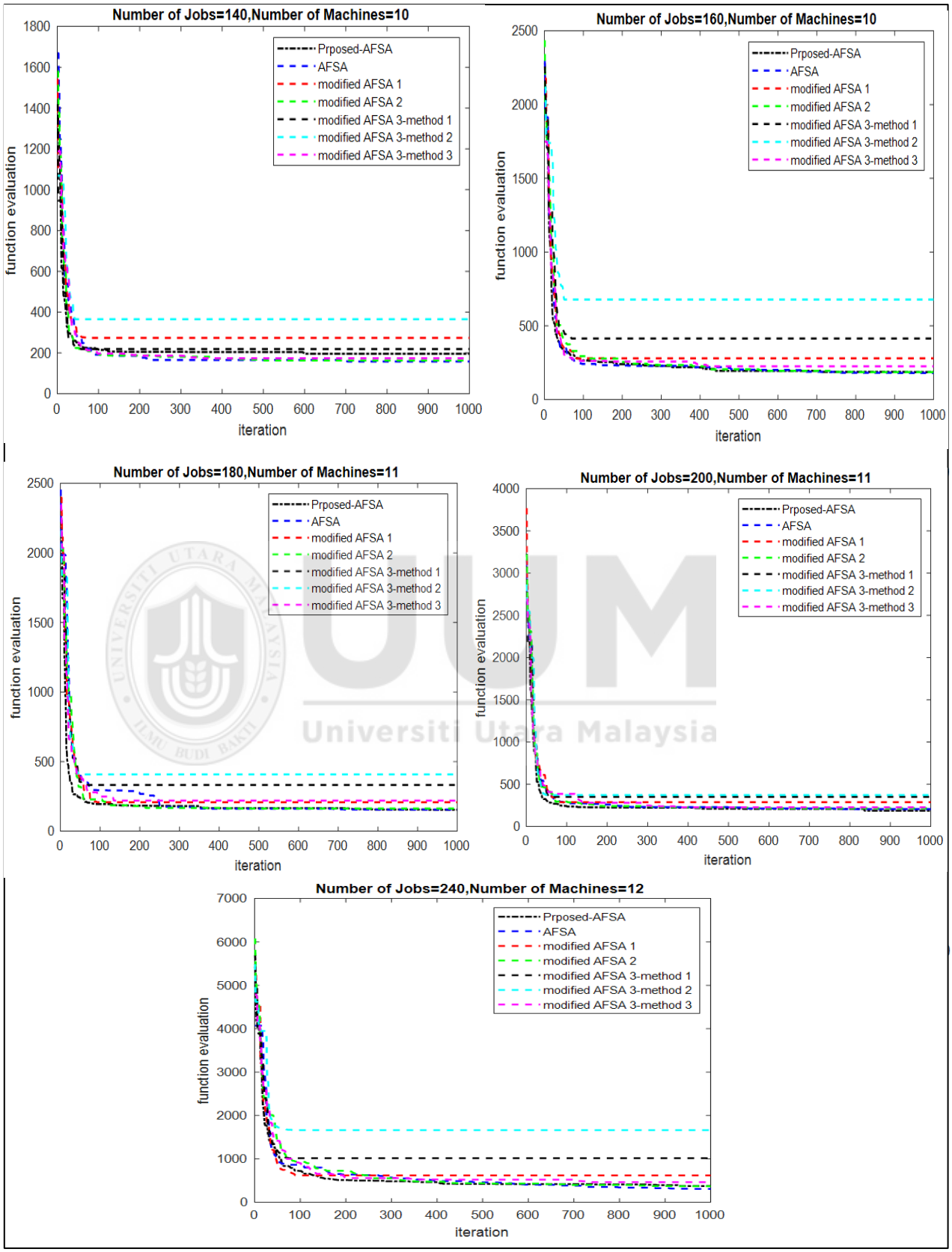
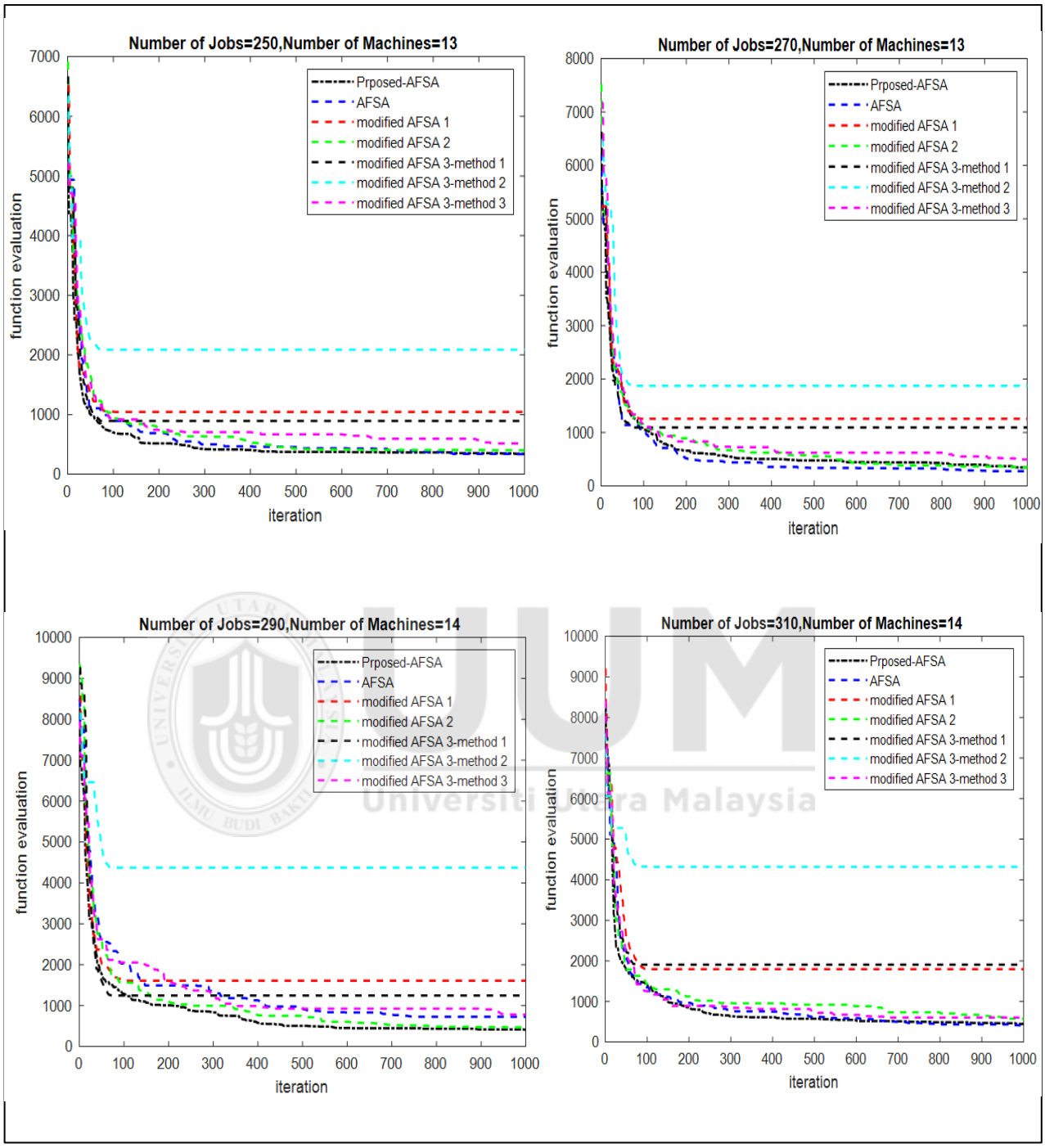


Figure B2: The performance of the proposed modified algorithm for meduim problem



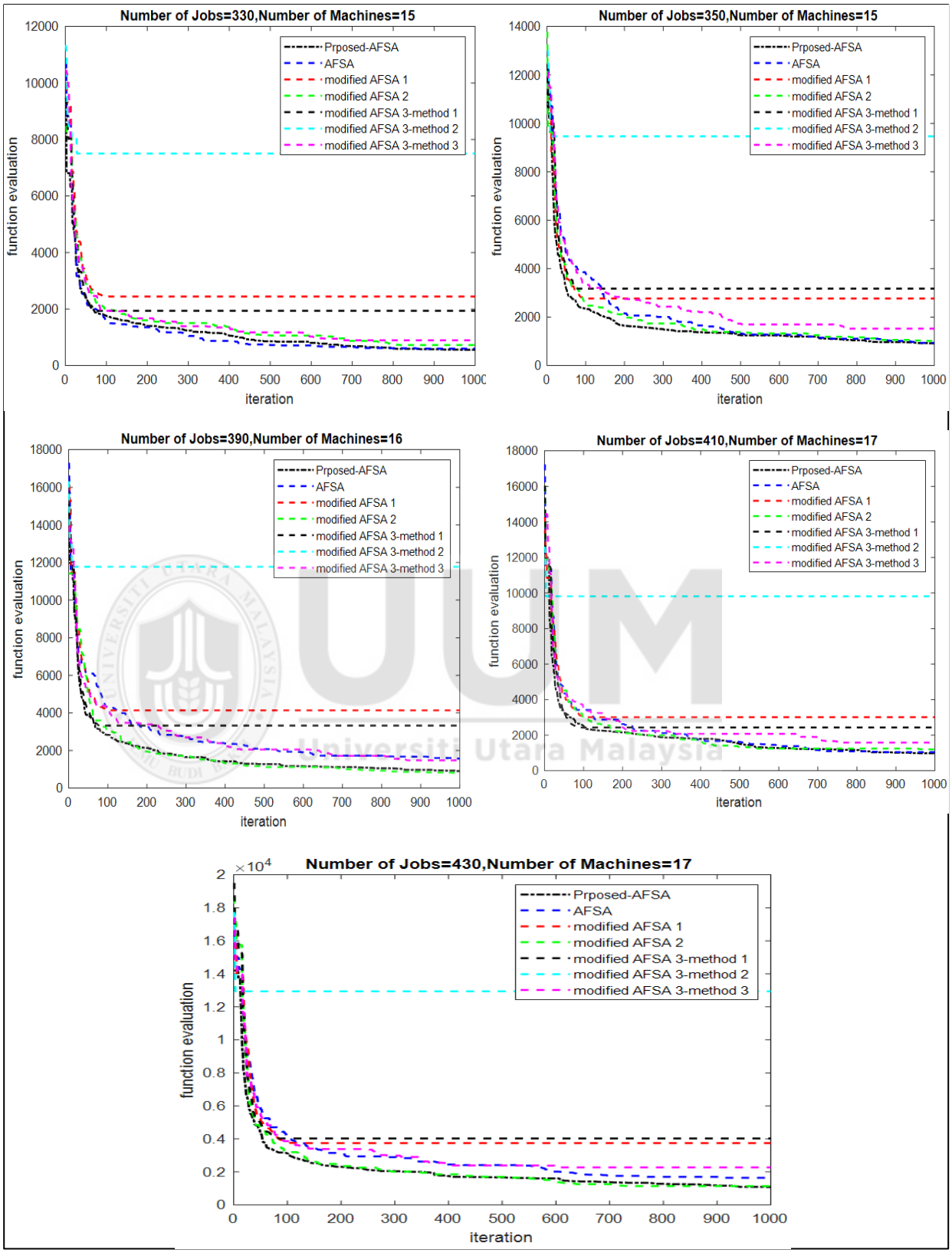
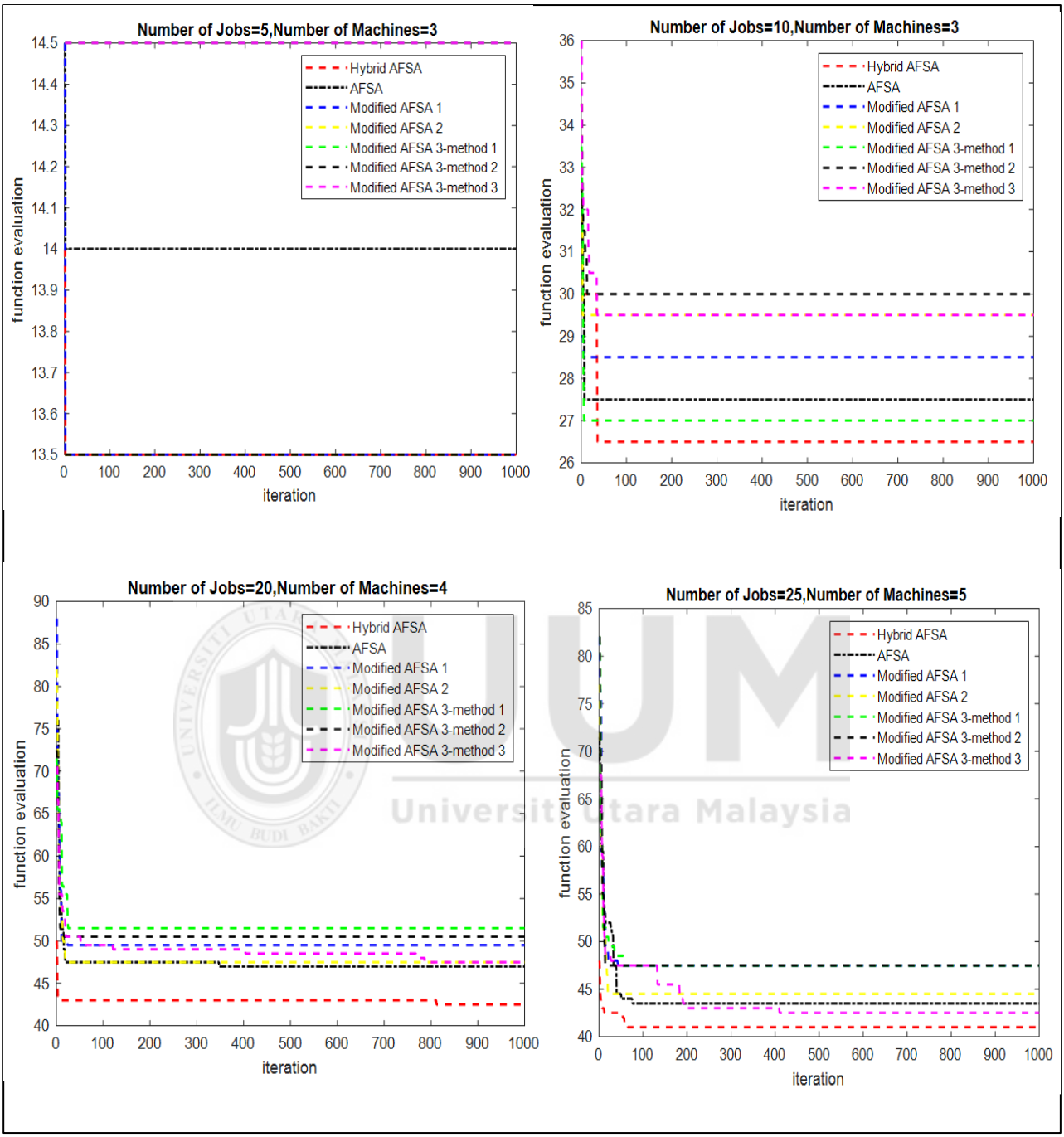


Figure B3: The performance of the proposed modified algorithm for large problem



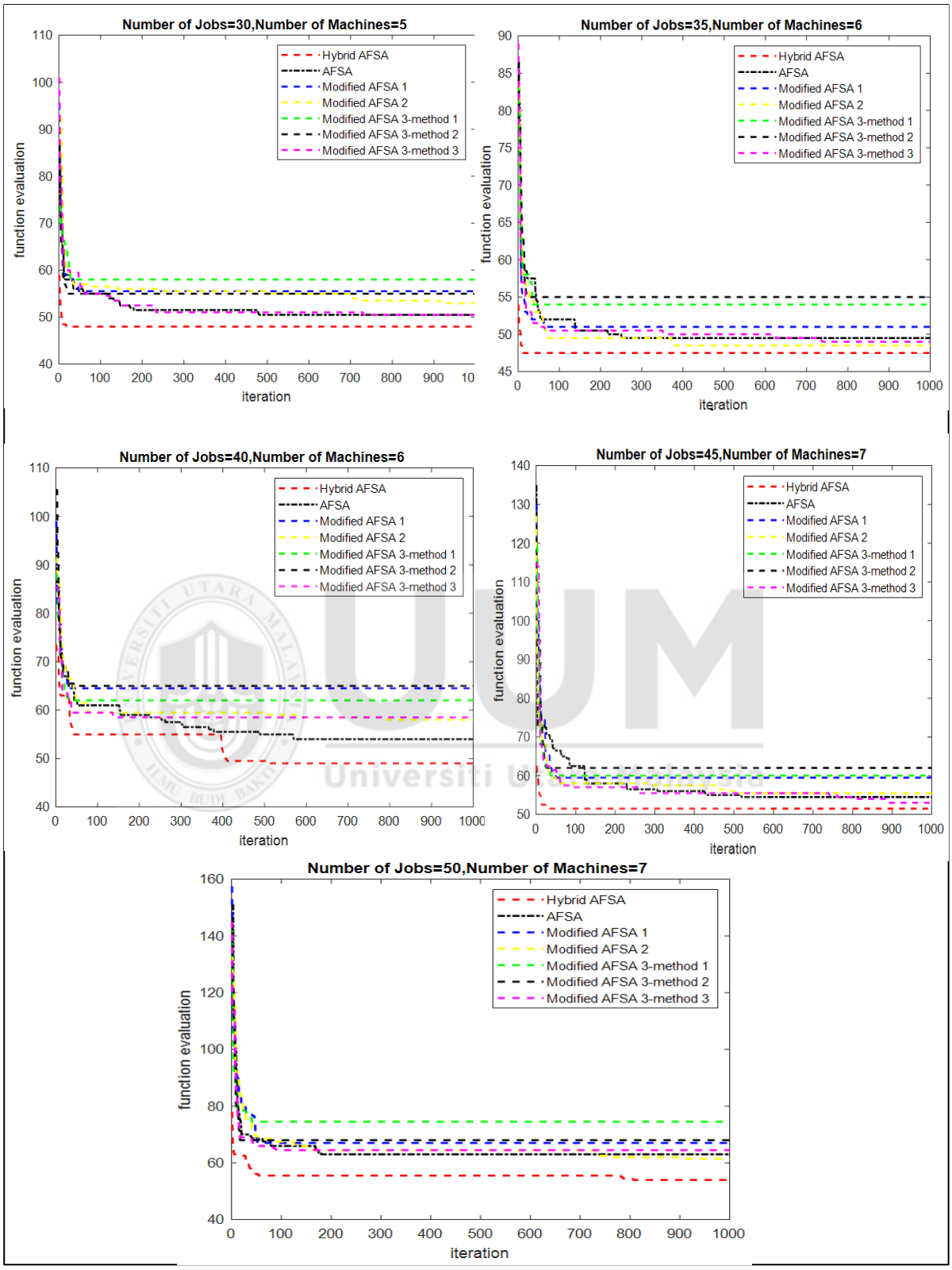


Figure B4: The performance of hybrid AFSA with standard AFSA and other modified AFSA algorithms for small-sized sample problems

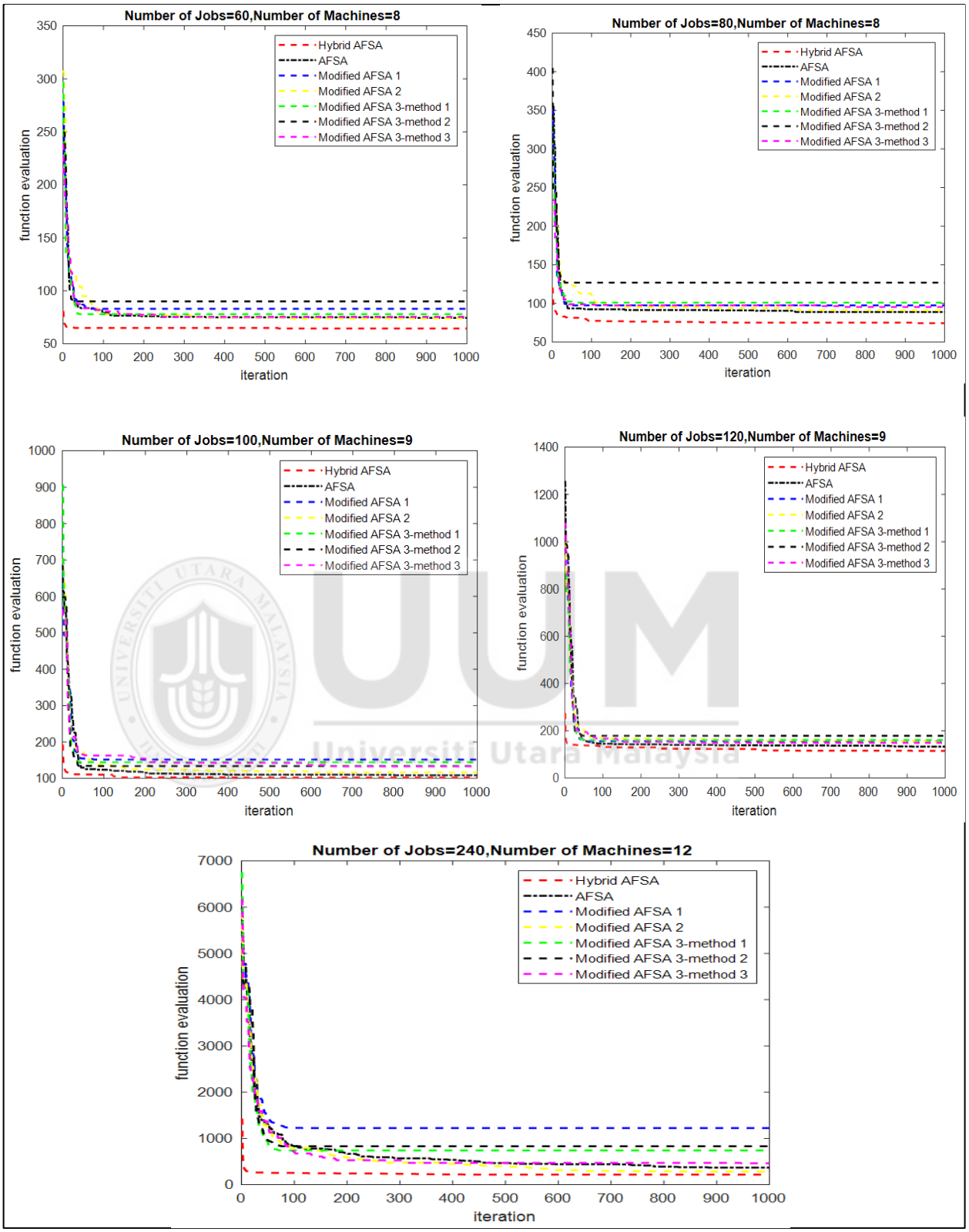
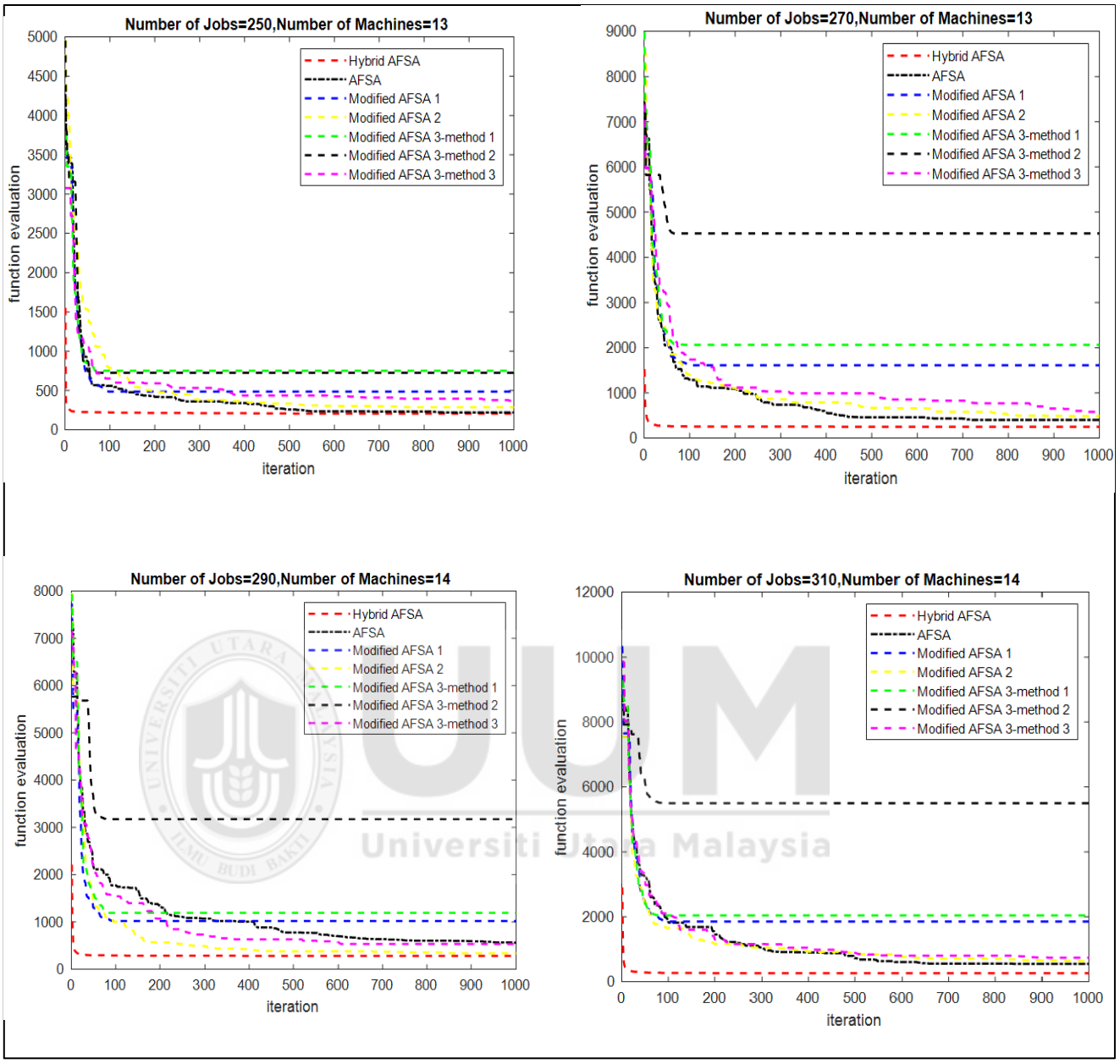


Figure B5: The performance of hybrid AFSA with standard AFSA and other modified AFSA algorithms for medium-sized sample problems.



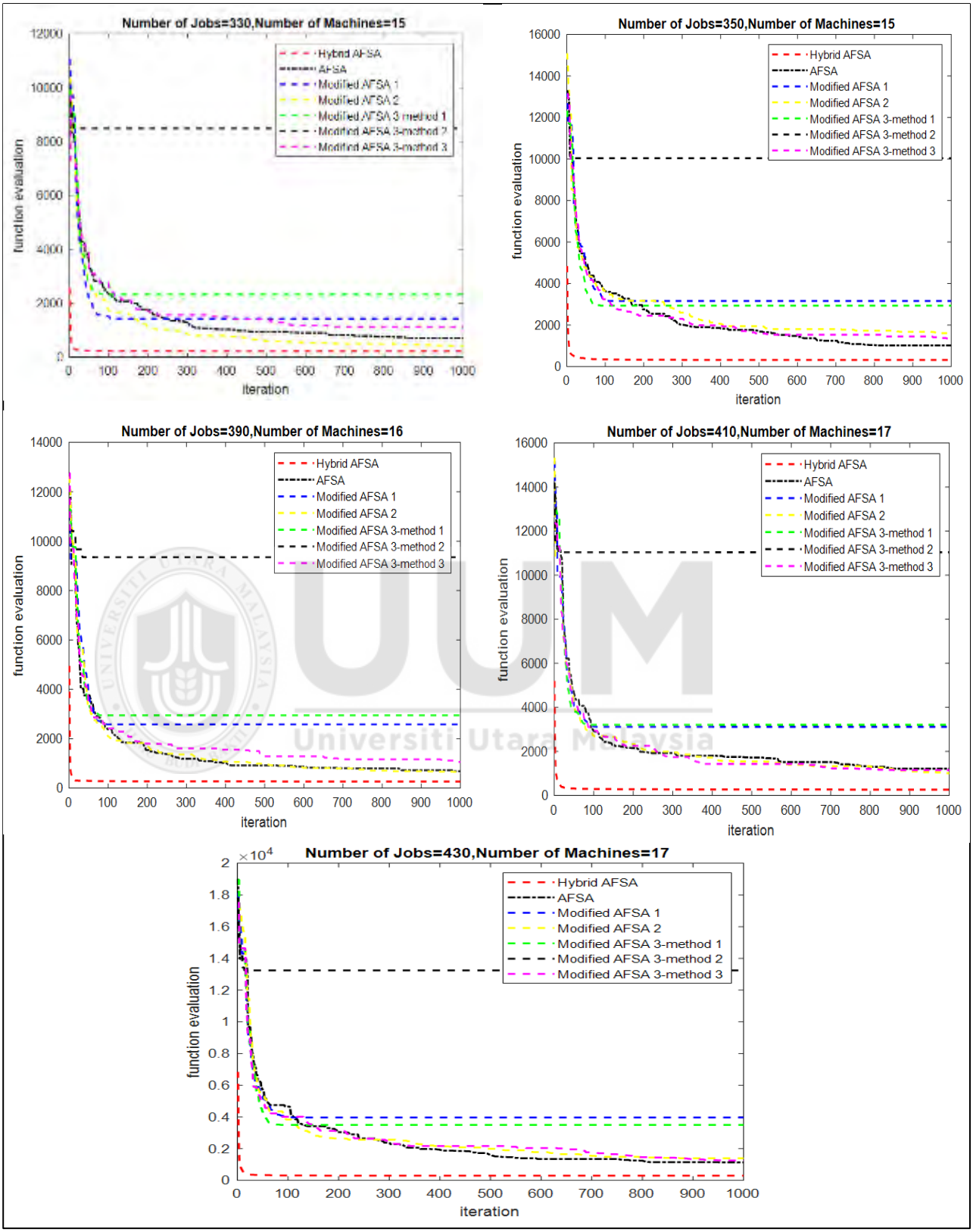
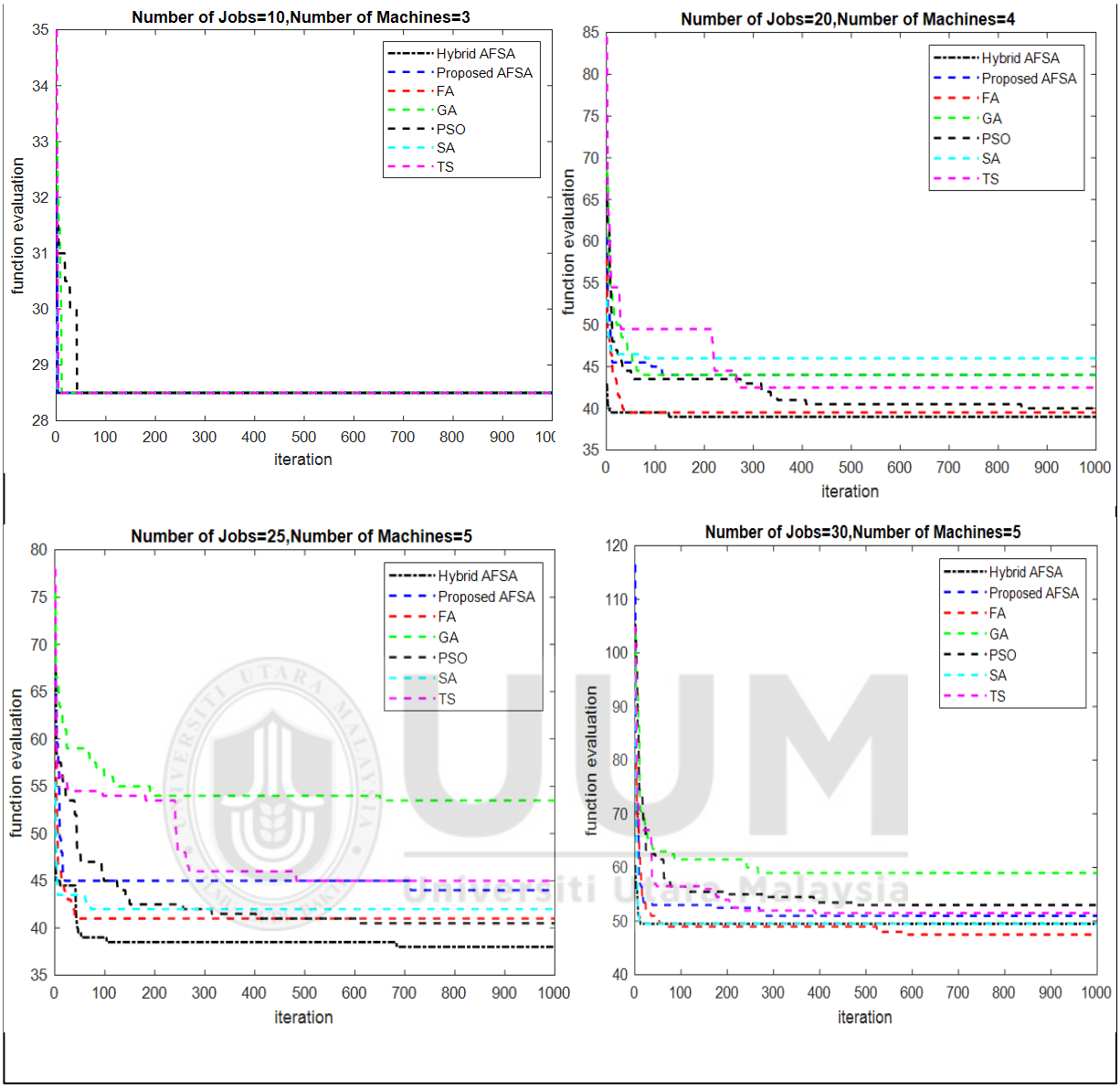


Figure B6: The performance of hybrid AFSA with standard AFSA and other modified AFSA algorithms for large sample problems



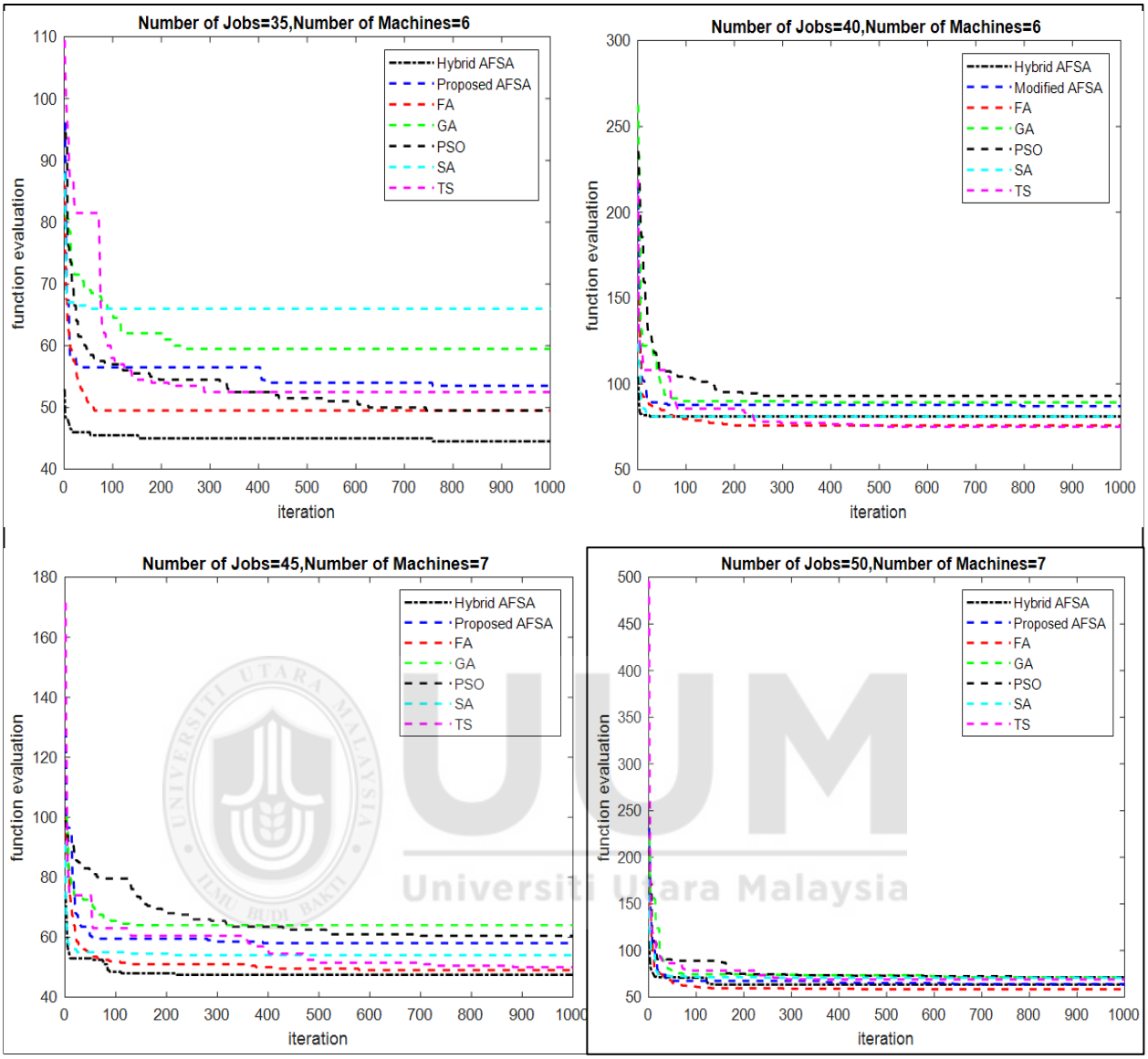
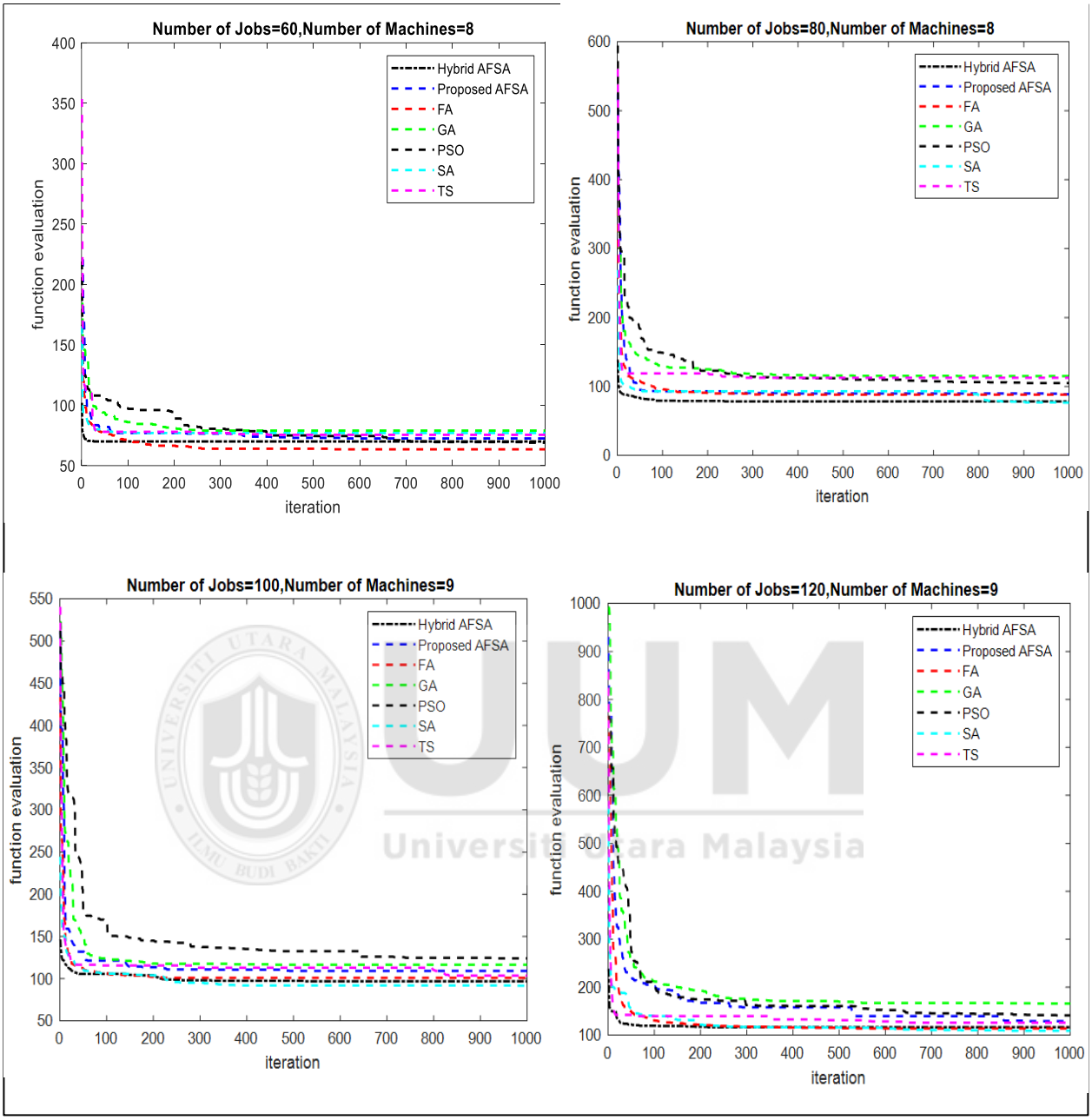


Figure B7: The performance of the hybrid algorithm for small-sized problems



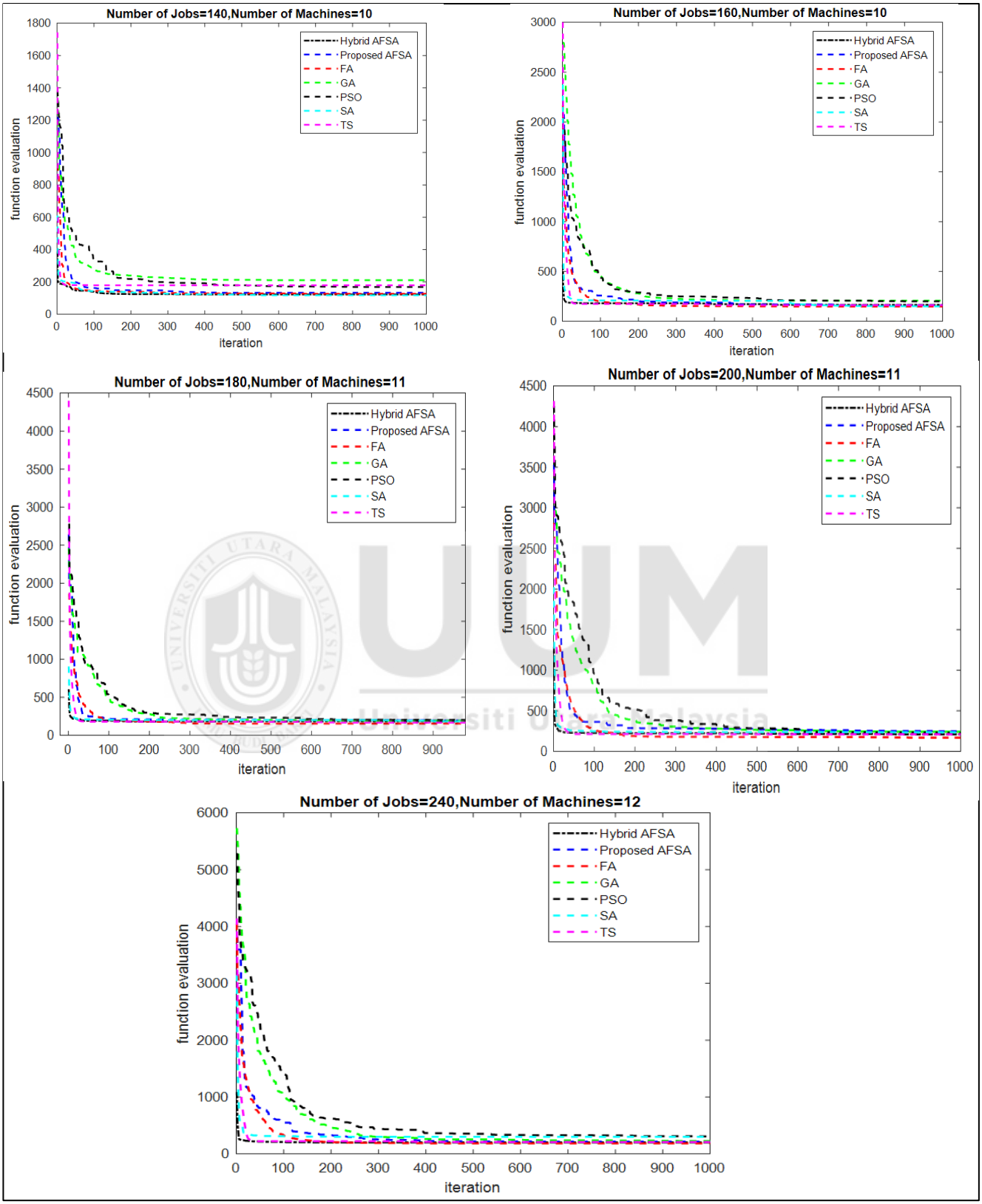
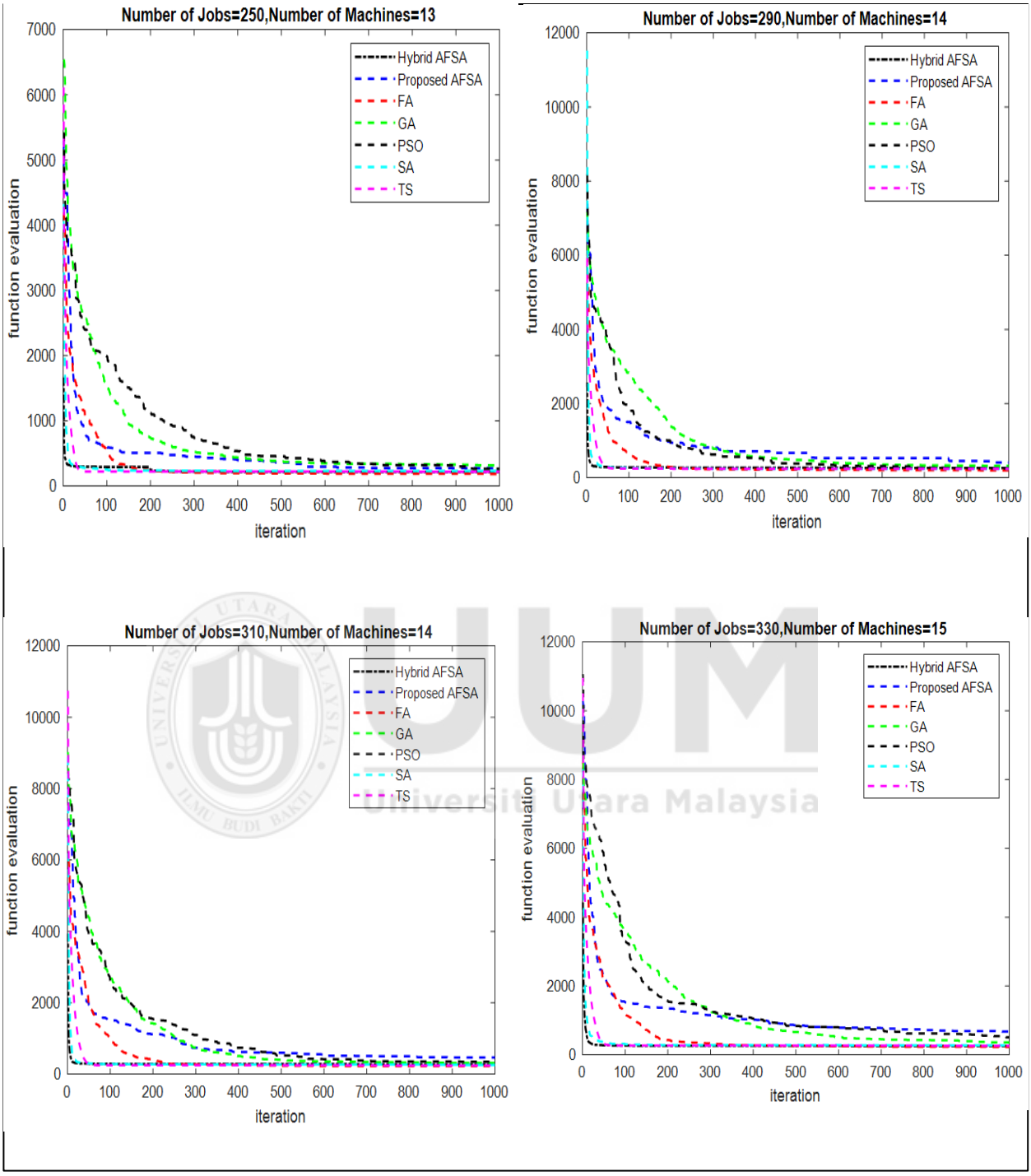


Figure B8: The performance of the hybrid algorithm for medium-sized problems.



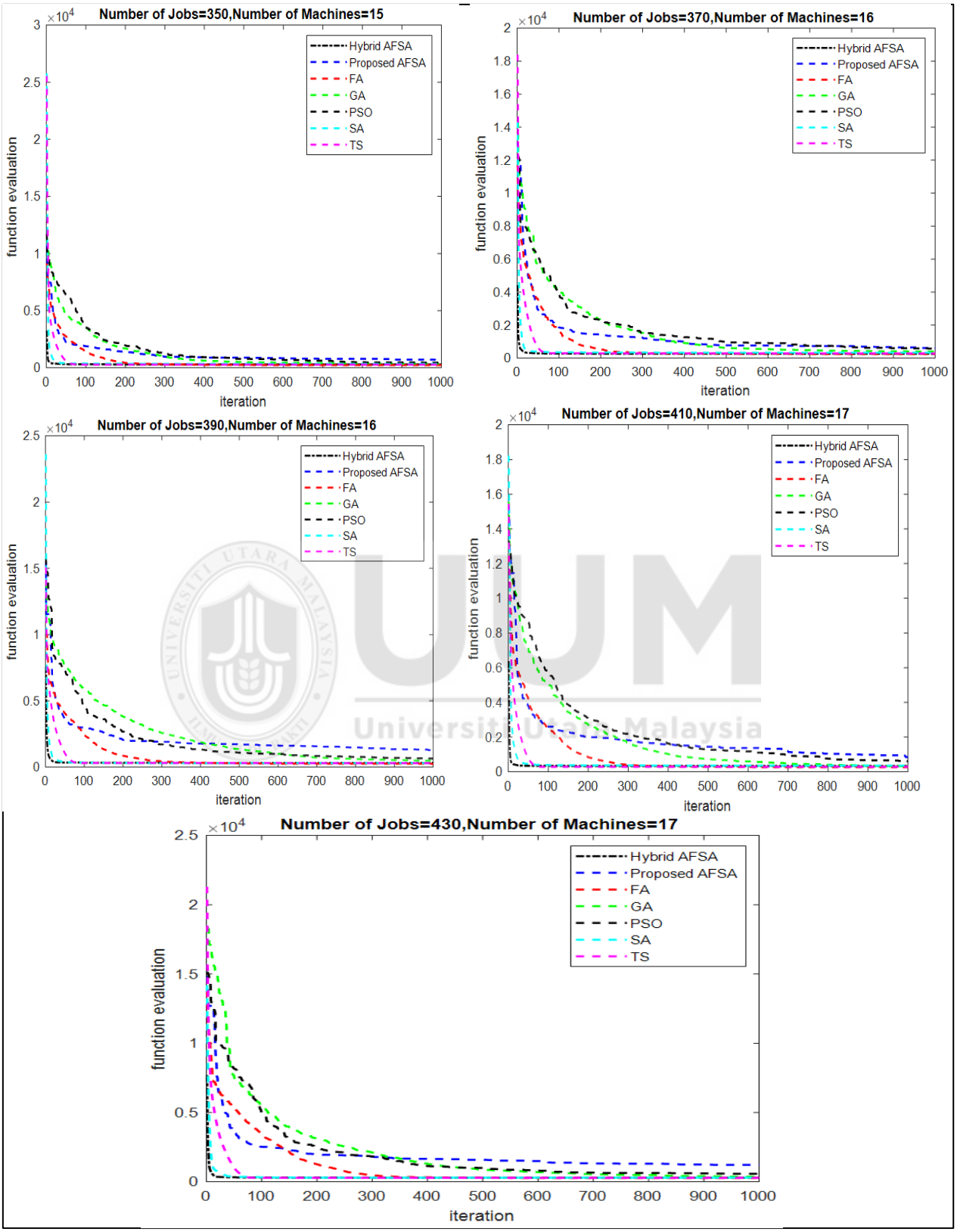
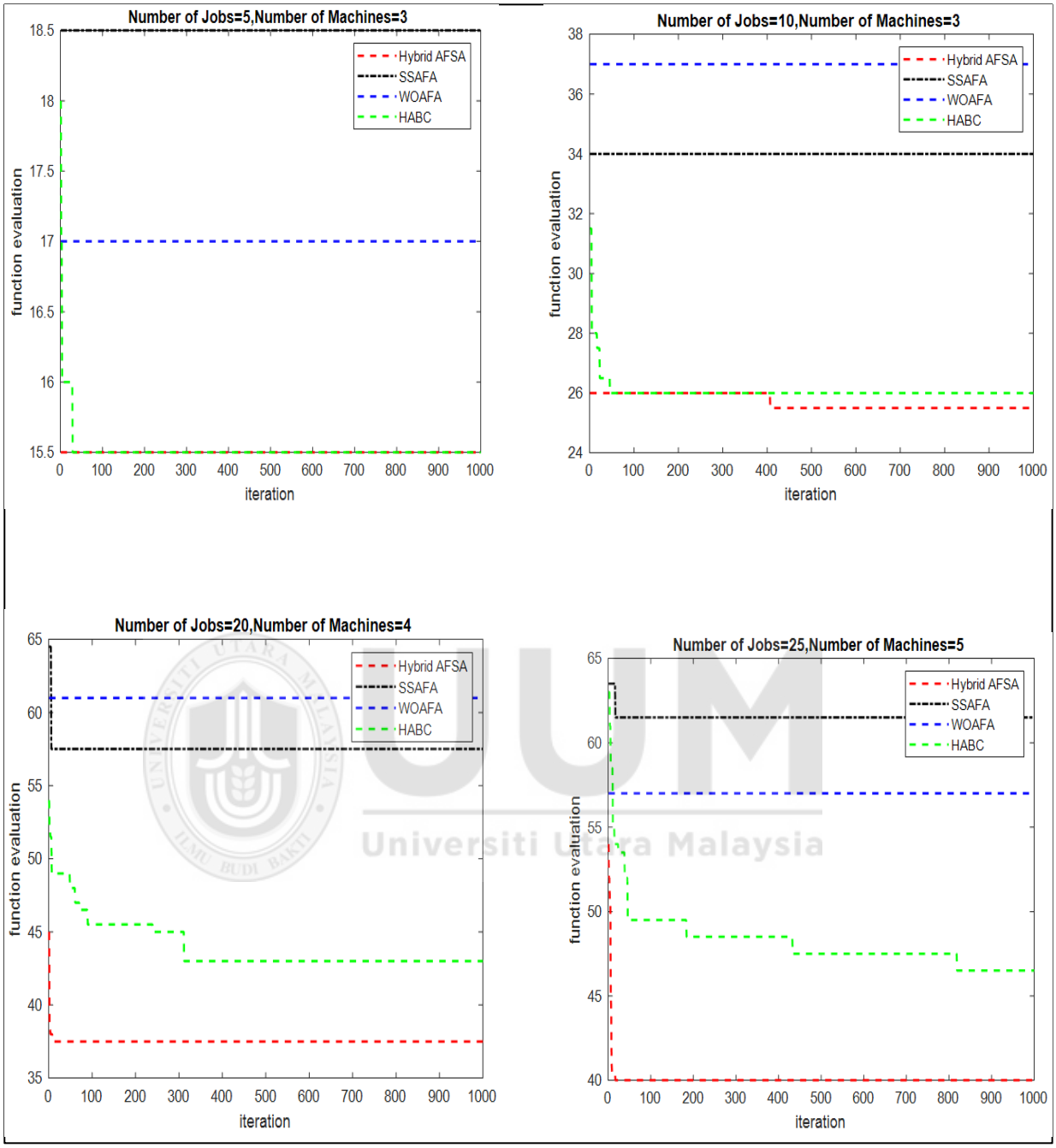


Figure B9: The performance of the hybrid algorithm for large-sized problems.



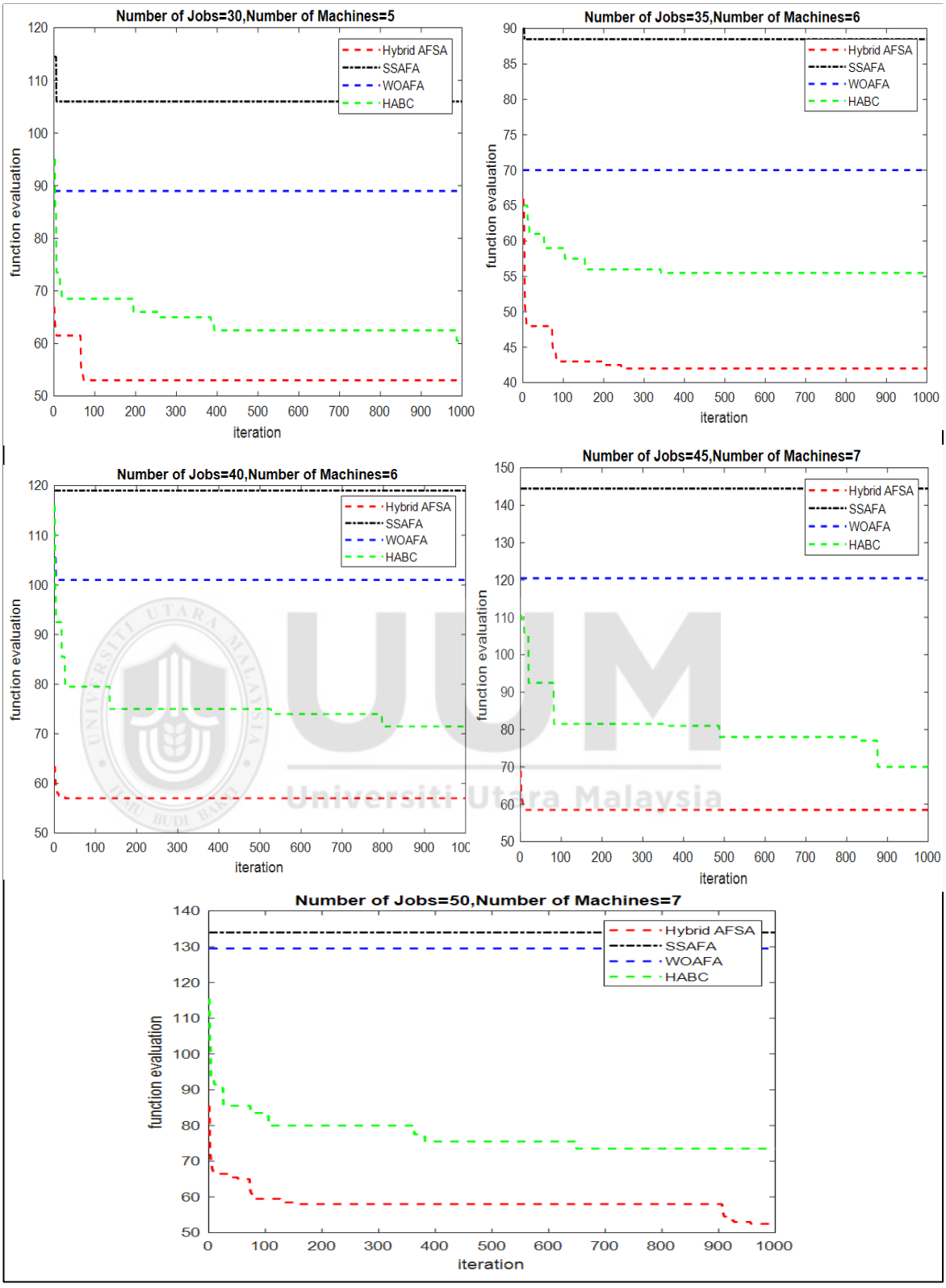
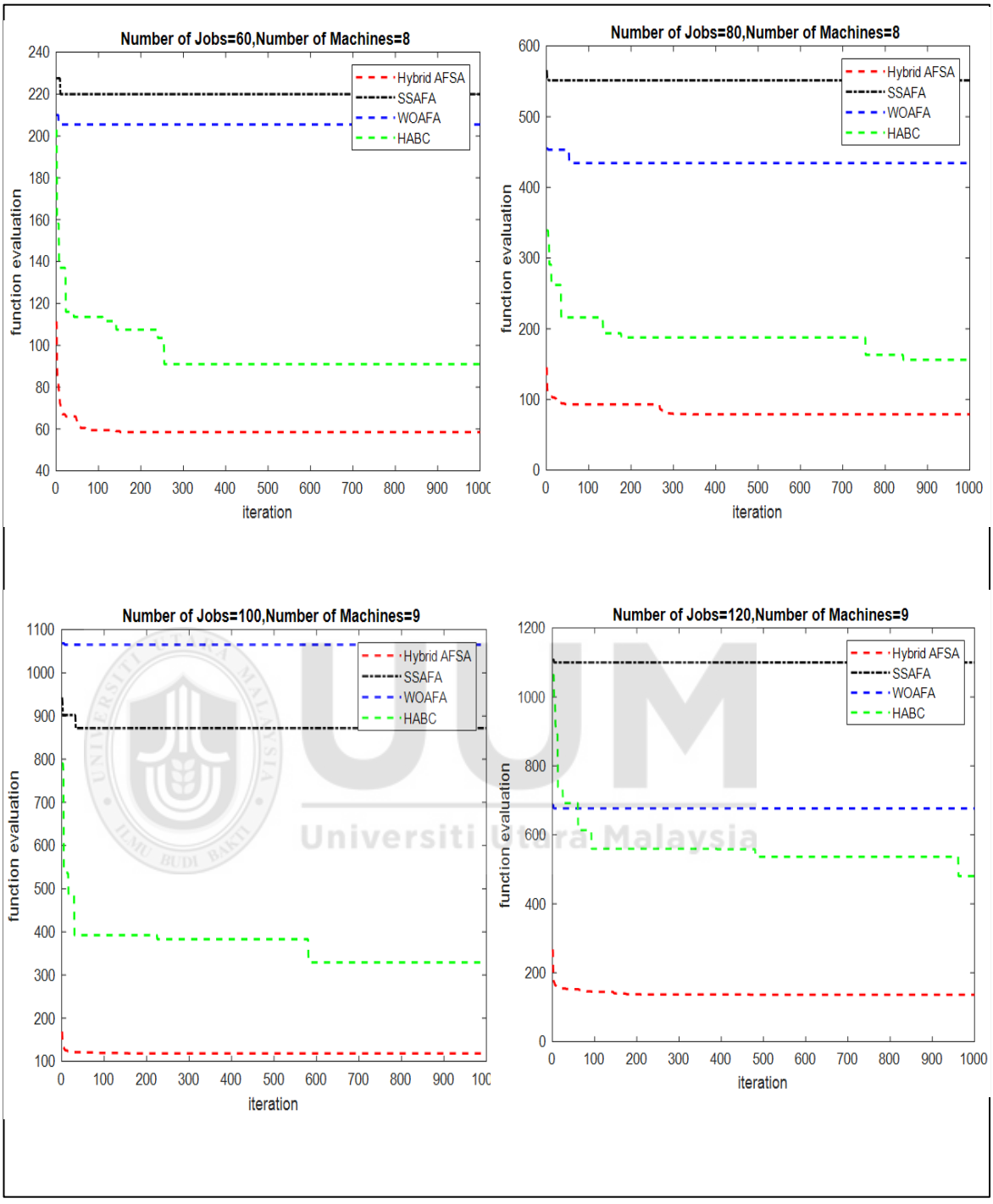


Figure B10: The performance of hybrid AFSA, HABC, WOAFA, and SSAFA for small-sized problems.



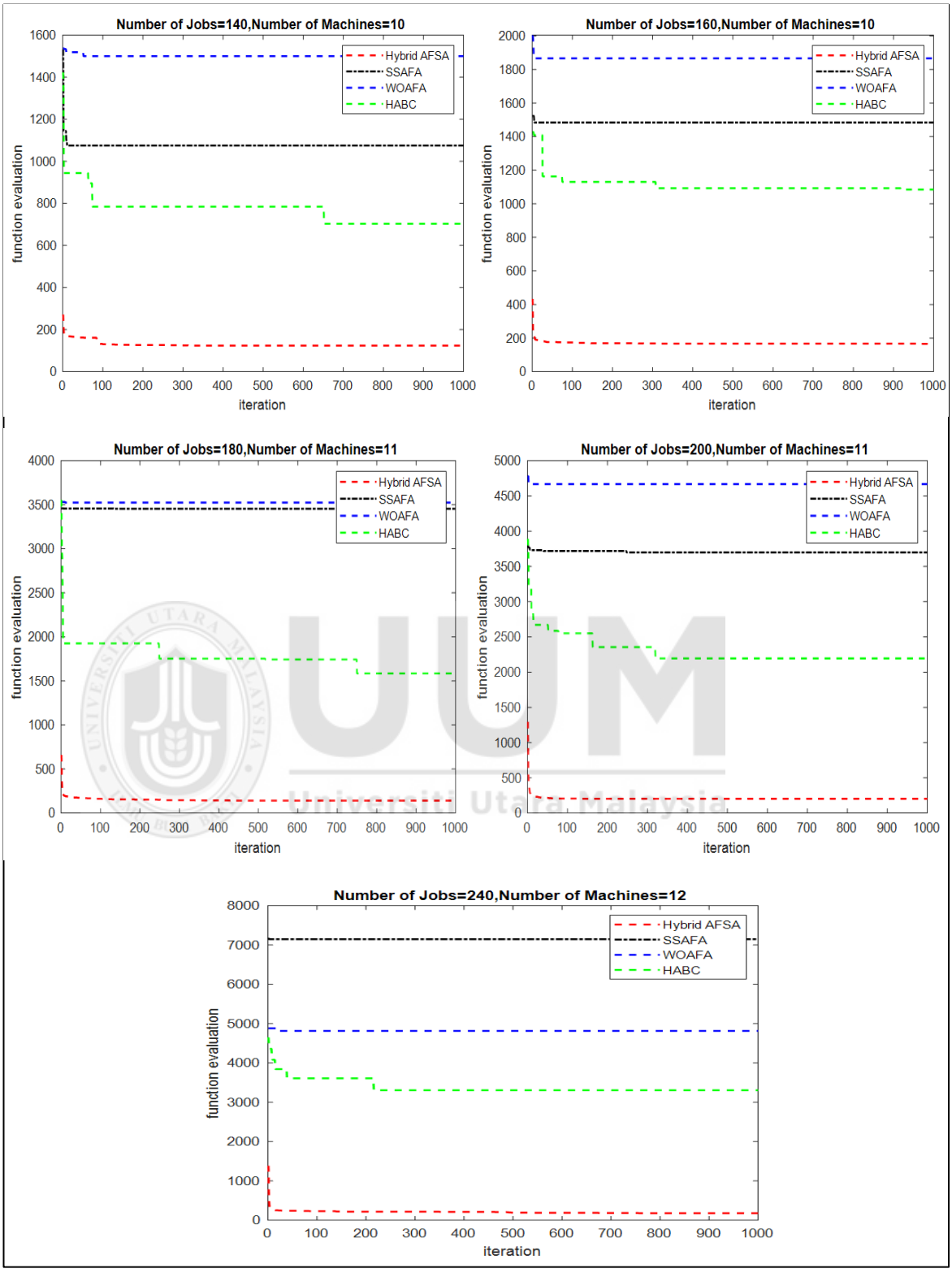
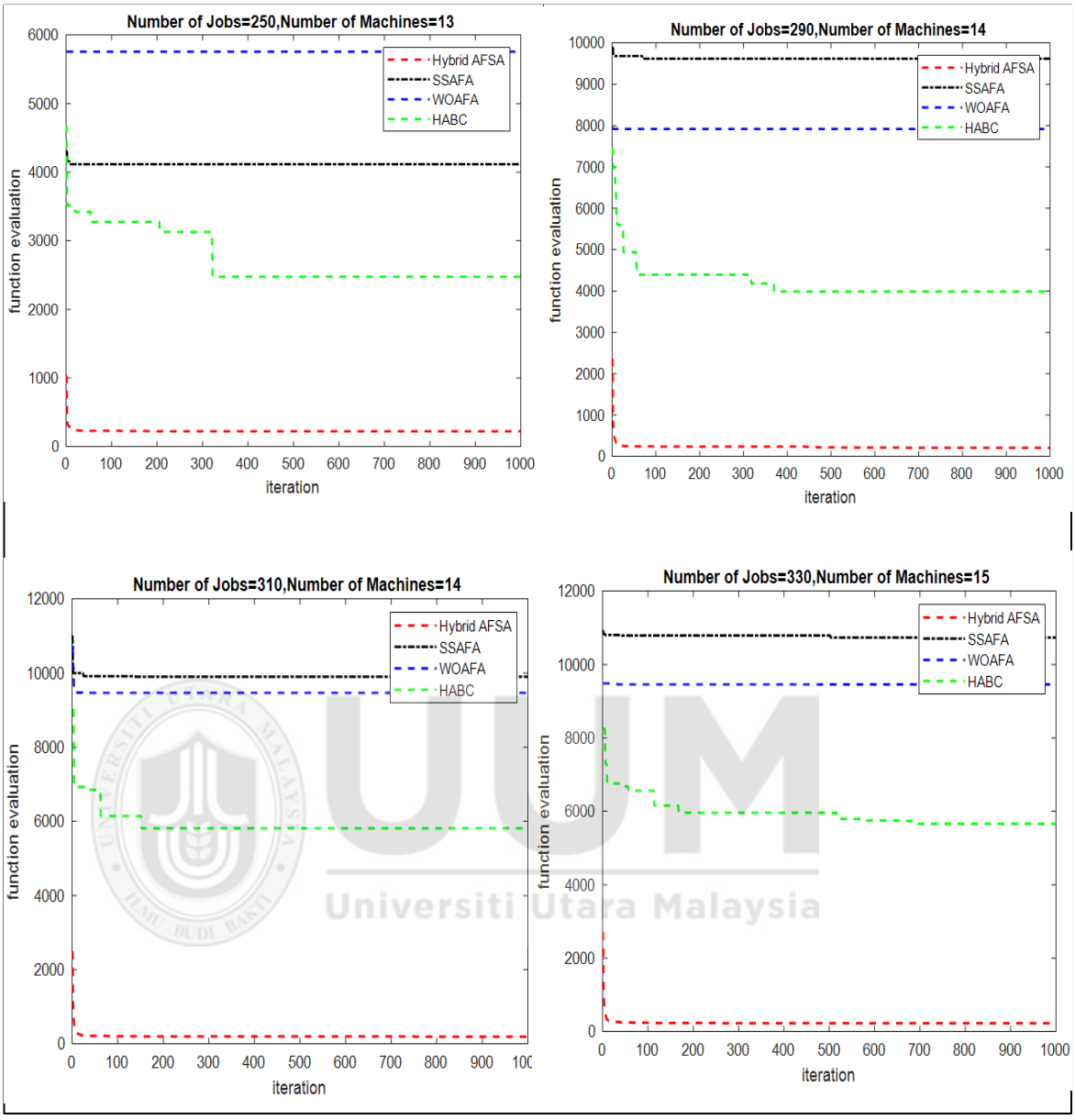


Figure B11: The performance of hybrid AFSA, HABC, WOAFA, and SSAFA for medium-sized problems.



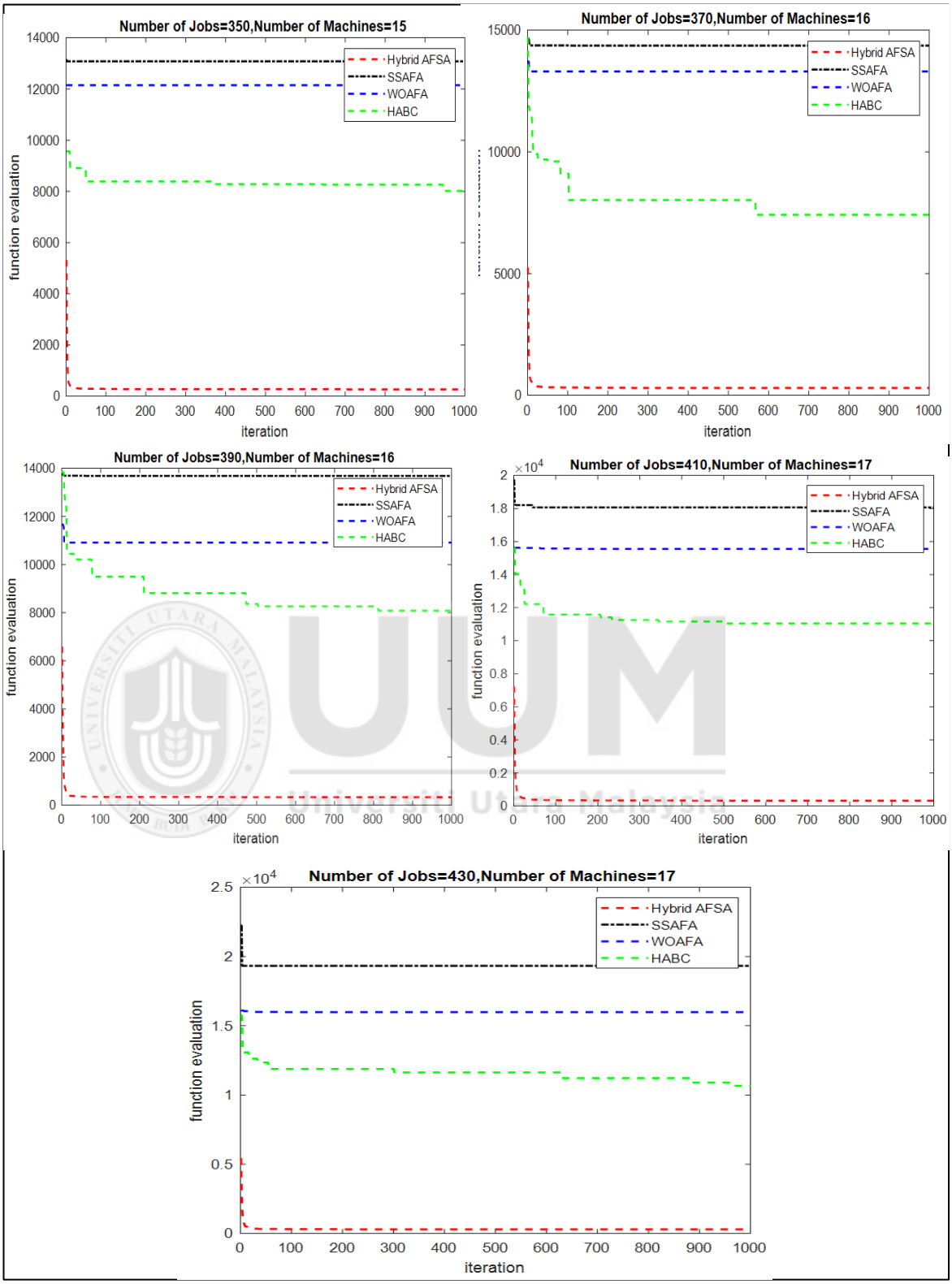


Figure B12: The performance of hybrid AFSA, HABC, WOAFA, and SSAFA for large-sized problems.