Journal of Intelligent Manufacturing (2022) 33:1939–1966
https://doi.org/10.1007/s10845-022-01940-1

Switching strategy-based hybrid evolutionary algorithms for job shop scheduling problems

Shahed Mahmud[1,2] · Ripon K. Chakrabortty[1] · Alireza Abbasi[1] · Michael J. Ryan[3]

Received: 18 August 2021 / Accepted: 17 March 2022 / Published online: 16 April 2022

Abstract
Since production efficiency and costs are directly affected by the ways in which jobs are scheduled, scholars have advanced a number of meta-heuristic algorithms to solve the job shop scheduling problem (JSSP). Although this JSSP is widely accepted as a computationally intractable NP-hard problem in combinatorial optimization, its solution is essential in manufacturing. This study proposes performance-driven meta-heuristic switching approaches that utilize the capabilities of multi-operator differential evolution (MODE) and particle swarm optimization (PSO) in a single algorithmic framework. The performance-driven switching mechanism is introduced to switch the population from an under-performing algorithm to other possibilities. A mixed selection strategy is employed to ensure the diversity and quality of the initial population, whereas a diversity check mechanism maintains population diversity over the generations. Moreover, a Tabu search (TS) inspired local search technique is implemented to enhance the proposed algorithm's exploitation capability, avoiding being trapped in the local optima. Finally, this study presents two mixed population structure-based hybrid evolutionary algorithms (HEAs), such as a predictive sequence HEA (sHEA) and a random sequence HEA (rHEA), and one bi-population inspired HEA, called bHEA. The comparative impacts of these varied population structure-based approaches are assessed by solving 5 categories of the standard JSSP instances (i.e., FT, LA, ORB, ABZ and TA). The performance of these hybridized approaches (i.e., sHEA, rHEA and bHEA) is compared and contrasted with its constituent algorithms (MODE, PSO and TS) to validate the hybridization's effectiveness. The statistical analysis shows that sHEA ranked first with mean value 1.84 compared to rHEA (1.96) and bHEA (2.21). Moreover, the proposed sHEA is compared with 26 existing algorithms and ranked first with a mean value 5.09 compared to the near-best algorithms. Thus, the simulation results and statistical analysis prove the supremacy of the sHEA.

Keywords Job shop scheduling problem · Multi-operator differential evolution · Particle swarm optimization · Evolutionary algorithm · Local search

Introduction

The job shop scheduling problem (JSSP) is a complicatedcombinatorial optimization problem (Sharma et al., 2018) that schedules a set of n jobs ($n \geq 1$) on a set of m machines ($m \geq 1$) under a set of constraints (Cruz-Chávez et al., 2019). Although there are many specific performance measures for JSSP (e.g. throughput time, tardiness, earliness, makespan and due-date), makespan is the most commonly used (Zhang et al., 2019). The JSSP has been proven in the literature to be a challenging NP-hard problem (Ibrahim & Tawhid, 2022) with $(n!)^m$ possible solutions, reflecting a high computational complexity (Zhao et al., 2018).

An extensive amount of research has been conducted to find the optimal or near-optimal solutions of this complex JSSP (Çaliş & Bulkan, 2015); however, the relatively small-sized 10×10 JSSP developed remained unsolved for a quarter of a century (Mishra et al., 2017). The realization of complexity and the incapability of exact approaches to solve even medium-sized JSSP has meant that research has focused on approximation approaches (Zhao et al., 2016). Many of these

✉ Shahed Mahmud
shahed.mahmud@student.adfa.edu.au

✉ Ripon K. Chakrabortty
r.chakrabortty@adfa.edu.au

✉ Alireza Abbasi
a.abbasi@unsw.edu.au

✉ Michael J. Ryan
mike.ryan@ieee.org

1 School of Engineering and IT, University of New South Wales, Canberra ACT 2620, Australia

2 Industrial & Production Engineering, Rajshahi University of Engineering & Technology, Rajshahi 6204, Bangladesh

3 Capability Associates, Canberra, Australia

approximation approaches include classical and emerging meta-heuristic approaches to be implemented to solve the JSSP, such as particle swarm optimization (PSO) and genetic algorithm (GA) (Zhao et al., 2016), differential evolution (DE) (Wisittipanich & Kachitvichyanukul, 2012), bacterial foraging algorithm (BFA) (Zhao et al., 2015), teaching-learning based optimization (TLBO) (Baykasoğlu et al., 2014), and grey wolf optimization (GWO) (Liu et al., 2020). However, these stand-alone algorithms could not find optimal solutions for many well-known JSSP instances (if known). The potential reason is that the application of these class of meta-heuristics in combinatorial optimization is challenging due to the lack of a fine-tuning property around optima and premature convergence (Zarandi et al., 2020), which encourages researchers to develop hybrid approaches for concurrently exploiting complementary features from two or more algorithms (Cruz-Chávez et al., 2019).

The DE algorithm is a population-based meta-heuristic approach that has successfully solved both continuous and discrete optimization problems. The most significant benefit of DE is that it has multiple mutation operators, and each has individual characteristics (Liu et al., 2020). However, Ponsich and Coello (2013) first implemented DE for JSSP and claimed that the selection operator of DE is essentially greedy, resulting in premature convergence and losing the opportunity to reach global optima. Ren and Wang (2012) also reported that DE prematurely converges because of filling similar individuals in the population over the evolution process, causing a lack of diversity in the population. Moreover, Ponsich et al. (2009) verified that DE alone is incapable of producing promising results over simple GA, greedy randomized adaptive search procedure (GRASP) or Tabu search (TS) in combinatorial optimization domains. On the other hand, PSO has been implemented successfully in different problem domains. However, Sha and Hsu (2006) conducted a comparative study for solving JSSP and concluded that stand-alone PSO could not produce optimal results for large-sized problems, but hybrid PSO could. The authors employed the TS in PSO, which results in better performance. The initial solution of PSO was improved by amalgamating a dispatching rule and a constructive heuristic, and the variable neighbourhood search was added to attain optimal solutions consuming less computational time (Marichelvam et al., 2020)–the authors claimed that hybrid PSO performs much better than PSO.

The population updating strategy of DE and PSO is distinct. To illustrate, the new swarm in PSO does not depend on whether the particle has been improved or not from the previous swarm (Wang et al., 2019) and, therefore, there is a chance of allowing the worst particle in the new generation. This concept is the opposite of DE, where only the improved individual is allowed for the next generation, which is why it is called a greedy selection process. This process can create a population with lower divergence, leading to premature convergence (Ponsich & Coello, 2013). These open problems are addressed by designing the DE with three popular mutation operators in which "DE/rand/1" and "DE/best/1" are well-known respectively for the exploitation property and exploration property, respectively and "DE/current-to-pbest/1" possesses both properties (Liu et al., 2020). In addition, this multi-operator DE (MODE) can be integrated with PSO, which may further lead to higher population diversity and ensure an improved evolutionary process, increasing the chance of achieving an optimal solution. An evolutionary process may perform poorly with the progression of generations, and it is inappropriate to allow the search method to continue the evolution (Elsayed et al., 2011). Thus, a population switching strategy (Wisittipanich & Kachitvichyanukul, 2012) based on the performance of an algorithm is employed in hybrid evolutionary algorithms (HEAs) to emphasize higher-performing algorithms. The lack of a fine-tuning property in meta-heuristics encourages the embedding of TS into this hybrid scheme to improve the local search property. TS is one of the strong local search techniques which has a strong exploitation property and can avoid becoming trapped in local optima (Sha & Hsu, 2006). Moreover, the initial population of this proposed algorithm is generated by implementing a mixed selection strategy (MSS) which considers both fitness value and the diversity of an individual while selecting schedules, since the initial population impacts the solution quality and the computational time (Cheng et al., 2016).

The major contributions of this research paper are:

- Diversity of the initial population is ensured by implementing an MSS considering both solution quality and diversity of a schedule.
- The proposed hybridization schemes of both MODE and PSO produce better-diverged populations with a better evolutionary process–to our knowledge, this is the first application of hybridizing MODE and PSO for these NP-hard problems.
- A performance-driven switching mechanism (PDSM) is implemented in the HEAs to escape the evolution process from the poorly performing search algorithm.
- Instead of ensuring diversity in personal best ($pbest$) and global best ($gbest$) only by considering makespan value as proposed by Sha and Hsu (2006), diversity in solution order of $pbest$ and $gbest$ is also ensured with makespan value to reduce unnecessary evaluations.

The performance of the predictive sequence HEA (sHEA), random sequence HEA (rHEA) and bi-population inspired HEA (bHEA) are evaluated along with stand-alone TS, DE and PSO for solving standard benchmark instances taken from the link http://jobshop.jjvh.nl/index.php. These are also

assessed against 26 published algorithms, including the most popular DE, GA and other recently developed algorithms. Parametric analysis is performed to set the best combination of parameters for the HEAs, and then statistical analysis is conducted for better insights into the results. The comparative analysis shows the dominance of sHEA over all other algorithms.

The remainder of this article is organized as follows: "Literature review" section describes the relevant literature review. "Constituent algorithms" section illustrates the fundamental of the constituent algorithms, after which the problem description of JSSP is given in "Problem description of a JSSP" section. The proposed algorithms and strategies developed in this study are explained in "Proposed algorithms" section, and "Experimental design and result analysis" section describes the experimental design and result analysis. Finally, concluding remarks and future directions of the research are presented in "Conclusion and recommendation" section.

Literature review

A range of influential studies can be found in the literature, concentrating on designing algorithms to solve the challenging NP-hard JSSPs. In early periods, classical techniques of operations research, which include, but are not limited to, the shifting bottleneck procedure (Adams et al., 1988), branch-and-bound technique (Brucker et al., 1994) and the dispatching rule approach (Kannan & Ghosh, 1993), were employed to solve JSSPs. These techniques can effectively solve small-sized instances. However, the complexity analysis of those techniques revealed that finding polynomial-time algorithms were challenging. Therefore, scholars and practitioners gradually began to pay attention to meta-heuristics, such as swarm intelligence (SI) and evolutionary algorithms (EAs). Recently, there has been a trend of exploiting and improving the SI, EAs and their hybridization to address JSSPs. This section aims to review the most relevant solution techniques of the JSSPs.

Wisittipanich and Kachitvichyanukul (2012) developed two DE algorithms for solving JSSP, considering two single objectives (i.e., makespan and total weighted tardiness). The effectiveness of these two algorithms was enhanced by dynamically balancing exploration and exploitation ability to avoid premature convergence. The first approach employed simultaneously different mutation strategies to compensate for the weakness of the individual strategy, whereas the second approach changed the search behaviour whenever the solutions did not improve. A local search was embedded to promote exploitation in the search space. These algorithms yielded promising results using less computational times and fewer function evaluations than the two-stage PSO (Pratchayaborirak & Kachitvichyanukul, (2011)). Investigating other mutation mixes and their robustness limit further implications to a broader range of scheduling problems. Baykasoğlu et al. 2014 implemented the TLBO algorithm to be testified its search mechanism in solving combinatorial optimization problems, adopting a random key based approach for obtaining a job permutation and Giffler and Thompson (G&T) algorithm (Giffler & Thompson, 1960) for active schedules. The authors claimed that this algorithm produced better results for a few test problems than some algorithms, such as hybrid GA (Ren & Wang, 2012), two memetic algorithms (Gao et al., 2011; Hasan et al., 2009) . Thus, further exploration is required to enhance its effectiveness in this domain. Qiu and Lau (2014) developed a hybrid artificial immune system (HAIS), where a modified PSO was employed to improve the antibody hypermutation process to accelerate the search procedure in solving 25 small-sized JSSPs. Thus, the algorithm's capabilities for solving complex JSSPs are required for further investigation. Wang and Duan (2014) claimed that the classic bio-inspired computational method, such as biogeography-based optimization (BBO), is incapable of solving the challenging NP-hard JSSPs, especially the large-sized instances. Thus, the authors developed a hybrid BBO (HBBO), employing the chaos theory and "searching around the optimum" strategy with the basic BBO. This hybridization expedited the convergence towards global optimum solutions. The hybridization's effectiveness was ensured compared with 14 popular algorithms, including modified PSO (Lin et al., 2010) in solving 44 instances. This study was limited to comparatively simple JSSPs, and thus the authors advised to consider novel techniques to enhance HBBO to handle more complicated problems. However, Lin et al. (2010) claimed that many algorithms based on heuristic algorithms, GAs, and PSO algorithms were implemented to solve JSSPs. Unfortunately, their results are not yet satisfactory.

Asadzadeh (2015) proposed an agent-based local search GA (aLSGA) for JSSPs by claiming that hybridization can enhance the performance and effectiveness of GAs. A multi-agent system that contains various agents, each with unique behaviours, was developed to implement the aLSGA. The experimental results showed its expedited convergence speed and improved solution quality. Zhao et al. (2015) developed a chemotaxis-enhanced bacterial foraging optimization (CEBFO) and added a DE operator, aiming at solving the tumble failure problem and accelerating the convergence speed of the original algorithm. An employed local search boosted exploitation capability. This algorithm proved its effectiveness in solving 38 instances against the popular hybrid PSO (Sha & Hsu, 2006) and TS guided shifting bottleneck (Pezzella & Merelli, 2000) and other classical algorithms. The authors claimed that transformation from continuous to discrete space requires much time and affects

the searching procedure. Akram et al. (2016) hybridized the fast simulated annealing (FSA) with quenching (HFSAQ), where the FSA performed global search and the quenching run for trajectory search. These dual roles prevented the algorithm from becoming trapped in local optima. This algorithm's effectiveness was established by solving 88 instances, among which 45 were solved optimally. This algorithm outperformed the 18 existing algorithms reviewed, including GA and TS (GATA) (Meeran & Morshed, 2014) that claimed that different features of many techniques are required to handle this challenging NP-hard problem. Kurdi (2016) developed an effective new island model GA (NIMGA) for JSSPs, minimizing the makespan. This study proposed a nature-inspired evolutionary model and a migration selection mechanism to improve search diversification and delay premature convergence. The evolutionary model employed different evolutionary methods while islands performed self-adaptation. The migration selection mechanism migrated worst individuals hoping to find a better chance to live in a more suitable environment. The author tested this algorithm's effectiveness in solving 52 instances; however, the global search capability can be further testified in solving complex JSSPs, such as Taillard instances (Taillard, 1993). Zhao et al. (2016) proposed a hybrid DE and estimation of distribution algorithm (EDA) based on a neighbourhood search (NS-HDE/EDA). The chaotic strategy was enhanced the searching ability, whereas the neighbourhood search improved the solution quality. EDA contributed to enhancing the global exploitation capability of the hybridization. Although this algorithm improved solution quality compared to standard GA and PSO, it was computationally expensive due to local search. Thus, designing an efficient local search is challenging but meaningful. Moreover, the greedy selection process reduces the population diversity (Ponsich & Coello, 2013).

Dao et al. (2018) developed a parallel bat algorithm (PBA), where random-key encoding scheme and communication strategy were employed. The PBA aimed to correlate individuals in swarms and share the computational load. The authors claimed that the communication strategy ensured the diversity-enhanced bats among the split population's groups to speed up solutions. The algorithm's effectiveness was tested against the classical bat algorithm and PSO (Ge et al., 2008) in solving 43 instances. Notably, Ge et al. (2008) improved the PSO's search capability using an artificial immune system (AIS). Jiang and Zhang (2018) improved the GWO algorithm by designing a discrete crossover operator, embedding an adaptive mutation method to keep population diversity and a local search for exploitation. The analysis showed that the discrete GWO outperformed the varied GWOs and many existing algorithms, including agent-based parallel GA (Asadzadeh & Zamanifar, 2010). Zhao et al. (2018) proposed a hybrid differential-based harmony search (DHS) algorithm as the population-based harmony search (HS) algorithm lacks local search capability. The authors employed the best individual in the pitch-adjustment process to expedite convergence and maintained a diversity of the population using the differential-based enhanced mechanism. Moreover, a modified variable neighbourhood search was employed to find solutions around the current harmony vector. Although this algorithm outperformed many state-of-the-art algorithms, the effectiveness of DHS is not very obvious while dealing with large-scale JSSPs. Thus, the authors suggested improving the DHS in this regard. Moreover, the proposed local search was computationally expensive, and thus further investigation was recommended. Although the artificial bee colony (ABC) algorithm efficiently solved many optimization problems, it required further improvement to solve the complex JSSP (Sharma et al., 2018). The authors maintained a proper harmony amid exploration and exploitation capabilities of the ABC by incorporating position update inspired by the beer froth (BeF) phenomenon, calling the algorithm BeFABC. The effectiveness of this algorithm was established against a list of existing algorithms, including parallel ABC (Asadzadeh, 2016); however, performance could be tested in solving complex machine scheduling problems.

Liu et al. (2020) claimed that WOA has already proved its effectiveness in solving a range of optimization problems. However, its performance was further enhanced with Lévy flight (LF) and DE (WOA-LFDE) to solve JSSP. The LF improved the abilities of global search and convergence in iteration, whereas the DE algorithm enhanced the local search and exploitation capabilities, keeping the diversity of solutions to escape local optima. The experimental results showed its superior performance against state-of-the-art algorithms, including HBBO (Wang & Duan, 2014), HAIS (Qiu & Lau, 2014), HFSAQ (Akram et al., 2016) and BeFABC (Sharma et al., 2018), in solving 88 instances. The authors directed to test the algorithm's performance in solving other combinatorial optimization problems. Mahmud et al. (2021) developed a two-step communication based EA, where MODE was employed at the beginning of the search to exploit the exploration capability of the population-based algorithm. The TS was employed later to utilize a local search capability. This study developed a decoding heuristic for an active schedule and enhanced the evolutionary process via mixed selection and communication strategies. This algorithm showed incapability in solving complex problems due to a lack of global search capability.

The articles reviewed above considered the JSSP as a challenging NP-hard problem in the combinatorial optimization domain, solving by developing a range of algorithms. The well-known and recently emerging algorithms were implemented since those algorithms have better exploration properties due to a multi-point searching capability (Zhao

et al., 2018). However, the exploitation incapability leads to higher computational expenses with poor solution quality (Zarandi et al., 2020) and the faster decline of diversity causes a premature convergence (Ren & Wang, 2012). In contrast, single-point search methods offer better exploitation; however, poor exploration and computational complexity limit their applications (Zhao et al., 2018; Liu et al., 2020). Thus, many hybrid approaches were developed, combining the first-rate features of multiple algorithms to optimize the trade-off between global search and local search capabilities (Mahmud et al., 2021), as reviewed above, explaining their strengths and drawbacks. However, the integration of multiple meta-heuristics into a single algorithmic framework can not be found in the production scheduling domain. Thus, this integration to enhance evolutionary performance can open up a new direction in combinatorial optimization problems. To fill the gap, this research proposes a unified framework of multiple meta-heuristics with the integration of switching strategy to emphasize the best performing meta-heuristic to continue the search process. This algorithm also includes a mixed selection strategy for the better initial population, a local search for exploitation property and a mechanism for ensuring population diversity over the evolutionary process. Finally, an extensive experiment is conducted by solving a wide range of standard JSSP instances.

Constituent algorithms

This section discusses the fundamentals of constituent algorithms, i.e., DE and PSO, of our proposed algorithms as a single-objective optimization problem under study.

Fundamental of a differential evolution (DE)

DE, which is one of the leading EAs in the literature, has already been applied in combinatorial optimization domains such as flow shop scheduling (Onwubolu & Davendra, 2006) and JSSP (Mahmud et al., 2021; Liu et al., 2009). This algorithm is prominent in the research community as it owns multiple variants, the ability to fast convergence, the simplicity to implement and more importantly, the capability of solving various optimization problems using the same parameter values (Sallam et al., 2020). Moreover, in literature, DE performed better than many other EAs such as GA for solving different optimization problems (Elsayed et al., 2012), and its mutation and crossover operators mainly control its performance (Ponsich & Coello, 2013). Although DE and GA are both variants of EAs, GA uses different solution updating mechanisms, including crossover, mutation, and elitism-preserving techniques. Crossover generates new solutions by exchanging genes among chromosomes, while mutation maintains diversity through small perturbation into the solutions to escape local optima, and elitism ensures the theorem "survival of the fittest". Although GA has been successfully implemented to handle many complex problems, it faces more difficulties than DE for handling multi-modal problems (Sallam et al., 2020).

Since, in this study, DE is the one main component algorithm, this section focuses on explaining DE's main steps. DE starts with initializing population and in the evolution process, includes mutation, crossover and selection. A minimization problem is considered in Eq. 1 to explained the DE, in which C_{max} denotes makespan of solution vector $\mathbf{x} = \{x_1, x_2, \ldots, x_D\}$, D is number of decision variables, and x_k^{min} and x_k^{max} indicate lower and upper bound of k^{th} decision variable.

$$\min C_{max} = \{f(\mathbf{x}) | x_k^{min} \leq x_k \leq x_k^{max}, \\ \forall k = 1, 2, \ldots, D\} \tag{1}$$

Population Initialization This algorithm initializes population of sized NP of D-dimension vector at generation $G = 0$, such as $Pop^o = [\mathbf{x}_{l,1}^o, \mathbf{x}_{l,2}^o, \ldots, \mathbf{x}_{l,k}^o, \ldots \mathbf{x}_{l,D}^o] | l = 1, 2, \ldots, NP$. Each $\mathbf{x}_l^G$ is known as target vector and the k^{th} position in l^{th} vector is filled using Eq. 2, where $rand(0, 1) \in [0,1]$ is a random probability distribution.

$$x_{l,k} = x_k^{min} + rand(0, 1) \times (x_k^{max} - x_k^{min}), \\ \forall l = \{1, 2, \ldots, NP\}; \forall k = \{1, 2, \ldots, D\} \tag{2}$$

Mutation In the evolution process, the first step is to generate a mutant vector ($\mathbf{M}$), where in variant $DE/rand/1$ as an example, three mutually exclusive candidate solutions ($\mathbf{x}_{r_1}^G \neq \mathbf{x}_{r_2}^G \neq \mathbf{x}_{r_3}^G$) are randomly picked from the current generation G and a mutant vector ($\mathbf{M}_l^G$) is generated by multiplying a scaling factor (F) to the differential vector of two target vectors ($\mathbf{x}_{r_2}^G$ and $\mathbf{x}_{r_3}^G$) and resultant is added to the third target vector($\mathbf{x}_{r_1}^G$) chosen, as presented in Eq. 3.

$$\mathbf{M}_l^G = \mathbf{x}_{r_1}^G + F \times (\mathbf{x}_{r_2}^G - \mathbf{x}_{r_3}^G) \tag{3}$$

F is a scale factor that leads to convergence speed and the population diversity (Liu et al., 2009).

Crossover In general, crossover is applied once mutant vector is generated. It generates an offspring solution known as a trial vector ($\mathbf{T}$) and could generally be binomial and exponential. Each decision variable k is considered in binomial operator if a generated random number is less than the crossover rate (C_r), as presented in Eq. 4.

$$\mathbf{T}_{l,k}^G = \begin{cases} \mathbf{M}_{l,k}^G & \text{if } rand(k) \leq C_r \text{ or } k = k_{rand} \\ \mathbf{x}_{l,k}^G & \text{otherwise} \end{cases} \tag{4}$$

In the above equation, $rand(k) \in [0, 1]$ and $k_{rand} \in [1, 2, 3, \ldots, D]$.

In exponential, a random integer d that defines the starting point taken from a target vector for crossover is chosen from $[1, D]$ and another integer b is chosen from $[d, D]$. It defines how many decision variables are selected from the donor solution. However, the trial vector is generated by Eq. 5, when both d and b are chosen.

$$\mathbf{T}_{l,k}^{G} = \begin{cases} \mathbf{M}_{l,k}^{G} & \text{for } k = \langle d \rangle_D, \langle c+1 \rangle_D, \langle c+d-1 \rangle_D \\ \mathbf{x}_{l,k}^{G} & \text{otherwise} \end{cases} \quad (5)$$

In equation, $\langle c \rangle_D$ represents a function of modulus D with starting point of d (Sallam et al., 2020).

Selection The produced trial vector ($\mathbf{T}_l^G$) is evaluated and compared with the target vector ($\mathbf{x}_l^G$) based on the fitness value. The trial vector is assigned to the target vector ($\mathbf{x}_l^{(G+1)}$) for the upcoming generation if the fitness value is lower in the trial vector for the minimization problem; otherwise, the current trial vector is mathematically copied into the next generation, as presented in Eq. 6.

$$\mathbf{x}_l^{(G+1)} = \begin{cases} \mathbf{T}_l^G, & \text{if } f(\mathbf{T}_l^G) < f(\mathbf{x}_l^G) \\ \mathbf{x}_l^G, & \text{otherwise} \end{cases} \quad (6)$$

Fundamental of a particle swarm optimization (PSO)

PSO is the population-based algorithm developed based on the social behaviour of the flocks of birds or the schools of fish, and it comprises a set of particles collectively called a swarm. A particle (which represents a job sequence) moves toward the personal best (**pbest**) obtained so far by itself and the global best (**gbest**) obtained by the swarm so far. The particle movement toward the **pbest** and **gbest** depends on the velocity, which is calculated using Eq. 7 while the position of a particle is subsequently updated using Eq. 8. $\mathbf{v}_l^G$ stands for velocity of particle l in generation G and $\mathbf{x}_l^G$ is the position of a particle l in generation G. w is the inertia and c_1 and c_2 control the movement of a particle towards the **pbest** and **gbest**, respectively. The $rand_1$ and $rand_2$ are variables having value in between 0 and 1.

$$\mathbf{v}_l^{(G+1)} = w \times \mathbf{v}_l^G + c_1 \times rand_1 \times (\mathbf{pbest}_l^G - \mathbf{x}_l^G) + c_2 \times rand_2 \times (\mathbf{gbest}_l^G - \mathbf{x}_l^G) \quad (7)$$

$$\mathbf{x}_l^{(G+1)} = \mathbf{x}_l^G + \mathbf{v}_l^{(G+1)} \quad (8)$$

Because of the simplicity and effectiveness of PSO, its many variants have been proposed to solve a range of changeling optimization problems, including in continuous (Wang et al., 2018) and discrete optimization (Sha & Hsu, 2006). Moreover, the PSO ensures a supreme quality of solution convergence (Qian & Li, 2018) and diversity (Islam et al., 2021), which is essential for such NP-hard problems under study.

Problem description of a JSSP

A JSSP comprises a finite set $J = \{J_1, J_2, J_2, \ldots, J_n\}$ of n non-homogeneous jobs indexed by j and a finite set $M = \{M_1, M_2, M_3, \ldots, M_m\}$ of m machines indexed by i. Each job j comprises a finite set $O_j = \{O_{j1}, O_{j2}, \ldots, O_{jm}\}$ of m tasks/operations indexed by t and can be processed by the m machines to complete the assigned work. Thus, sequencing and processing the n jobs to the m machines are the typical aim of any $n \times m$ JSSP, considering different objective functions (Wang et al., 2018; Zhang et al., 2019). However, the sequence of operations on one job should be predefined. Each operation $O_{jt} \in O_j$ can be processed by one of m machines and neither this operation nor the assigned machine will be interrupted by any other job until the current job's work is finished. It means that no preemption is allowed and the assigned machine are considered available (Liu et al., 2020). Total operations are $n \times m$, excluding the additional two dummy operations at the start and end with zero processing time. $C_j = max\{C_{j1}, C_{j2}, \ldots, C_{jm}\}, \forall j$ indicates job completion time and the makespan is calculated by setting $C_{max} = max\{C_1, C_2, C_3, \ldots, C_n\}$.

The constraints which must be followed while building a JSSP are as follows:

- A job cannot be processed more than once on the same machine.
- A job must satisfy its precedence relations, if any.
- Each machine can process only one job at a time.
- A job cannot be processed in multiple machines at the same time.
- Release time and due dates are not specified.

The generated solution of this problem is a schedule, which is a sequence of operations to machines with deterministic processing times. This complex combinatorial NP-hard problem is optimized in this study, minimizing the makespan value.

Proposed algorithms

As previously described, since no single algorithm and/or search strategy can solve a wide range of optimization problems, multi-operator and/or multi-method based approaches have been emerging to handle this difficulty. In this work, MODE and PSO algorithms are proposed and implemented in a single algorithmic framework with distinct multi-populated strategies to solve a wide range of JSSPs. This

section presents the proposed algorithms (i.e., sHEA, rHEA and bHEA).

bHEA

The idea of simultaneous multi-search is implemented in the proposed hybrid optimization framework bHEA, in which the equally divided sub-populations evolve in MODE and PSO and exchange information to strengthen the searching power (Cruz-Chávez et al., 2019). The quality of the evolution process of both algorithms are measured, and if one algorithm performs better than another, the best performing algorithm continues the search process until both become under-performing. Then, the sub-populations are switched between these two algorithms after enhancing the quality of **pbest** and **gbest** through evolving in a local search. The Algorithm 1 presents the main steps of the proposed bHEA.

Algorithm 1 The framework of the proposed bHEA

1: Define NP; $FES \leftarrow 0$; MAX_{FES}; $Count_1 \leftarrow 0$; $Count_2 \leftarrow 0$; $G \leftarrow 0$, $PreMax$;
2: Generate an initial population of size $4 \times NP$ as described in Sect. 5.3;
3: Convert the generated population to preference list as explained in Sect. 5.4 and calculate fitness value using Algorithm 3;
4: Compute goodness score using Equation 11 and select initial population of size NP with higher score as illustrated in Sect. 5.5;
5: Initialize the **pbest** and **gbest**;
6: Set $FES \leftarrow (FES + 4 \times NP)$;
7: Create two sub-populations for two search algorithms $\{SP_{alg|alg=1,2}\}$ of size $\{nAlg_1 = nAlg_2 = \frac{NP}{2}\}$ and ; Divide SP_1 into three sub-SP for three MODE variants $\{SSP_{op|op=1,2,3}\}$ of each size $\frac{nAlg_1}{3}$;
8: **while** $FES \leq MAX_{FES}$ **do**
9: **for** $l = 1 : \frac{NP}{2}$ **do**
10: **if** $l \leq \frac{nAlg_1}{3}$ && $Count_1 \leq PreMax$ **then**
11: Generate mutant vector, $\{\mathbf{M}^G_{l,op|op=1,2,3}\}$ from SSP_1,SSP_2 and SSP_3 using Equations 12, 13 and 19, respectively;
12: Generate trial vector, $\{\mathbf{T}^G_{l,op|op=1,2,3}\}$ of the $\{\mathbf{M}^G_{l,op|op=1,2,3}\}$ using Equation 4;
13: **end if**
14: **if** $Count_2 \leq PreMax$ **then**
15: Update velocity vector $\mathbf{v}^{G+1}_{lik}$ and position of a solution order vector $\mathbf{S}^{G+1}_{lik}$ corresponding to $\mathbf{x_l}$ in SP_2 using Algorithm 4;
16: **end if**
17: **end for**
18: Update SP_1 using Equation 6 and Algorithm 5, set $Count_1 \leftarrow (Count_1 + 1)$ using Equation 20; regroup $\{SSP_{op|op=1,2,3}\}$;
19: After evaluating SP_2, update solution using Algorithm 5 and set $Count_2 \leftarrow (Count_2 + 1)$ using Equation 20;
20: **if** $Count_1 == Count_2 == PreMax$ **then**
21: Apply local search (Algorithm 6) on a randomly selected **pbest** from $\{SP_{alg|alg=1,2}\}$ and update solutions using Algorithm 5;
22: Apply population switching; Create the $\{SSP_{op|op=1,2,3}\}$ from SP_1 as explained before; finally, set $Count_1$ and $Count_2 \leftarrow 0$;
23: $FES \leftarrow (FES + FES_{LS})$;
24: **else if** $Count_1 || Count_2 == PreMax$ **then**
25: $FES \leftarrow (FES + \frac{NP}{2})$;
26: **else**
27: $FES \leftarrow (FES + NP)$;
28: **end if**
29: $G \leftarrow (G + 1)$;
30: **end while**

Initially, a population of $4 \times NP$ candidate solutions are generated using the Latin Hypercube Design (LHD), as it efficiently covers the search space (Sallam et al., 2017), as described in the Sect. 5.3. Then, the generated continuous random solution strings are converted to integer strings as explained in the Sect. 5.4 and then, a population of size NP is chosen with giving priority to both diversity and fitness value in the selection procedure, known as MSS, to ensure better-searching experience and to avoid premature convergence (see the Sect. 5.5). The population is then divided into two sub-populations, i.e., SP_1 and SP_2, to receive benefits from multi-method. The SP_1 is further grouped into three, i.e., $SSP_{op|\{op=1,2,3\}}$, and searches in parallel with chosen operators of DE. The choices are made meticulously to ensure diversity and convergence simultaneously, as illustrated in Sect. 5.6.1. In order to avoid premature convergence, a regrouping is introduced in each generation (Tasgetiren & Suganthan, 2006). The performance of each evolution method is measured based on the capability of updating **gbest** (line 18 and 19 in Algorithm 1 and Eq. 20). If either MODE or PSO fails to improve the **gbest** to a specified number of times ($PreMax$), the corresponding evolution process is halted until the same scenario arises in another one. Then, a proportion of **pbest** is sent to a local search to overcome the problem created in MODE-PSO since local search is more capable of producing solutions around local optima than MODE and PSO. It works by creating neighbour solutions of the candidate pool. The **pbest** and **gbest** are updated in every generation in such a way that each of **pbest** will have a distinct makespan and solution order to assure better evolution of PSO, described in Algorithm 5. Whenever the local search terminates, the sub-populations of MODE and PSO switch between them. The population switching between algorithms solely depends on the performance of the search process, and therefore it is named PDSM. This HEA is named bHEA since it is designed based on the bi-population concept. The bHEA terminates when the current number of fitness evaluations (FES) reaches the maximum number of fitness evaluations (MAX_{FES}).

rHEA and sHEA

Population is commonly believed to be one of the most crucial features of EAs and enables explorations to different parts of the search space via a set of individuals. Thus, dividing the population into smaller sub-populations may lose search uniformity and exploration. However, having a larger population size may not always be helpful (Chen et al., 2012). Thus, this study, as shown in Algorithm 2, integrates both scenarios in a single algorithmic framework through a population switching strategy. If any algorithm shows an adequate performance over the generations, no other algorithms evolve to solve the problem. Otherwise, the population among the algorithms switches depending on the performance measure.

In Algorithm 2, the randomly generated population is filtered based on the fitness value and the diversity of a schedule by employing a developed MSS, as described in Sect. 5.5. The initial population is then divided into three smaller sub-populations, i.e., $SP_{op|\{op=1,2,3\}}$ of each sized $\frac{NP}{3}$ before evolving in the MODE, in which each sub-population uses their mutation and the defined crossover. Over the evolution process, the algorithm may perform poorly, and it is inappropriate to carry out the search process (Elsayed et

Algorithm 2 The framework of the proposed sHEA (rHEA)

Define NP; $FES \leftarrow 0$; MAX_{FES};$Count_1 \leftarrow 0$; $Count_2 \leftarrow 0$; $G \leftarrow 0$; $PreMax$;
Generate an initial population of size $4 \times NP$ as described in Sect. 5.3;
Convert the generated population to preference list as explained in Sect. 5.4 and calculate fitness value using Algorithm 3;
Compute goodness score using Equation 11 and select initial population of size NP with higher score as illustrated in Sect. 5.5;
Initialize the $pbest$ and $gbest$;
Set $FES \leftarrow (FES + 4 \times NP)$;
Create three sub-populations $\{SP_{op|op=1,2,3}\}$ of each sized $\frac{NP}{3}$ for MODE operators ; Set $Sel \leftarrow 1$;
while $FES \leq MAX_{FES}$ **do**
  if $Sel == 1$ **then**
    for $l = 1 : \frac{NP}{3}$ **do**
      Generate mutant vector, $\{\mathbf{M}^G_{l,op|op=1,2,3}\}$ from $\{SP_{op|op=1,2,3}\}$ using Equations 12, 13 and 19, respectively;
      Generate trial vector, $\{\mathbf{T}^G_{l,op|op=1,2,3}\}$ of the $\{\mathbf{M}^G_{l,op|op=1,2,3}\}$ using Equation 4;
    end for
    Update $\{SP_{op|op=1,2,3}\}$ using Equation 6 and Algorithm 5; set $Count_1 \leftarrow (Count_1 + 1)$ using Equation 20; regroup $\{SP_{op|op=1,2,3}\}$;
  else if $Sel == 2$ **then**
    for $l = 1 : NP$ **do**
      Update velocity vector $\mathbf{v}^{G+1}_{lik}$ and position of a solution order vector $\mathbf{S}^{G+1}_{lik}$ corresponding to $\mathbf{x_l}^G$ using Algorithm 4;
    end for
    After evaluating, update solution using Algorithm 5 and set $Count_2 \leftarrow (Count_2 + 1)$ using Equation 20;
  end if
  if $Count_1 == PreMax$ **then**
    Apply local search (Algorithm 6) on a randomly selected **pbest** from $\{SP_{op|op=1,2,3}\}$ and update solution using Algorithm 5;
    $Sel \leftarrow 2$ (**for rHEA, randomly assign** 1 **or** 2 **to** Sel); $Count_1 \leftarrow 0$; $FES \leftarrow (FES + FES_{LS})$;
  else if $Count_2 == PreMax$ **then**
    Apply local search (Algorithm 6) on a randomly selected **pbest** and update solution using Algorithm 5;
    $Sel \leftarrow 1$ (**for rHEA, randomly assign** 1 **or** 2 **to** Sel); $Count_2 \leftarrow 0$; $FES \leftarrow (FES + FES_{LS})$;
  else
    $FES \leftarrow (FES + NP)$;
  end if
  Set $G \leftarrow (G + 1)$;
end while

al., 2011). Thus, the population is switched from an underperforming algorithm to another, emphasizing the better performing algorithm. A performance indicator that counts the inability of the current search process to improve the global best (**gbest**) solution over the evolution process is designed to control the switching decision and compared with the $PreMax$. The algorithm with low performance quickly reaches $PreMax$, and then the population switches to another algorithm. If the MODE can satisfy the population switching condition, a certain proportion of **pbest** is sent to a local search. Similarly, the local search performance is measured and compared with Tabu termination criteria. The inferior performance of the local search allows the entire population to select the PSO that evolves using its operator and follows the same termination criteria of the MODE. The population switching between algorithms solely depends on the performance of the search process. This HEA is called sHEA as it follows a specific sequence. Instead of switching population between algorithms sequentially, the implemented PDSM with a random algorithm selection is developed, named rHEA and highlighted in Algorithm 2.

Population initialization

The LHD is employed to generate an initial distinct continuous-based population as it is capable of producing more efficiently scatter points across the solution space (Salam et al., 2017).

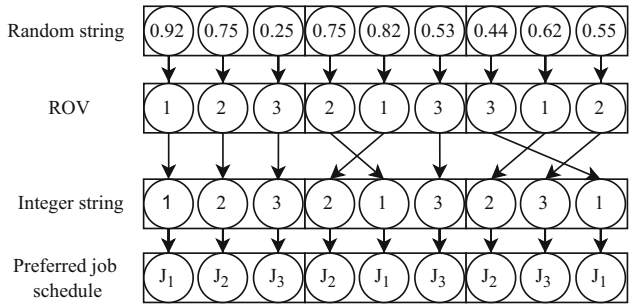


Fig. 1 The mapping of the algorithm from continuous to discrete

$$x_{l,k} = x_k^{min} + (x_k^{max} - x_k^{min}) \times lhd(1, NP)$$
$$\forall l = \{1, 2, \ldots, NP\}, \forall k = \{1, 2, \ldots, D(= n \times m)\} \quad (9)$$

lhd in Eq. 9 is a function of the LHD to produce random real numbers.

Solution representation, encoding and decoding schemes

Solution representation has a significant impact on designing algorithms and achieving the highest degree of performance (Cheng et al., 2016). Ponnambalam et al. (2001) compared among several solution representations (e.g. operation-based representation, job-based representation, preference-based representation, and priority rule-based representation) and concluded that the preference-based representation provides the best solution quality, which is therefore implemented in this study. An example of $n \times m$ JSSP is considered to illustrate the representation, where a string consists of m sub-strings, each for one machine. Each sub-string length is n, and each number in a sub-string identifies an operation that has to be processed on the relevant machine. The sub-string only prescribed the preference list of the corresponding machine, not the actual operation sequence.

To illustrate the preference-based representation with encoding, a random string is [(0.92, 0.75, 0.25), (0.75, 0.82, 0.53), (0.44, 0.62, 0.55)], where each string consists of $m = 3$ sub-string of dimension $1 \times n$ ($n = 3$), as depicted in Fig. 1. The values in each sub-string are real numbers generated at random in interval [0,1]. These real numbers are then transformed into an integer series based on ranked-order-value (ROV) (Zhao et al., 2018), which results in integer string [(1, 2, 3), (2, 1, 3), (2, 3, 1)] known as the preference list. The first sub-string (1, 2, 3) is the preference sequence of machine M_1, the second sub-string (2, 1, 3) is for machine M_2, and the third sub-string (2, 3, 1) is for machine M_3. An active schedule, where no operation can be drawn into without delaying any other operations (Ahmadian et al., 2021), is

Table 1 Data set for 3×3 JSSP

Job	Machine sequence			Job processing time		
	O_1	O_2	O_3	M_1	M_2	M_3
1	M_1	M_3	M_2	5	4	6
2	M_1	M_2	M_3	3	6	3
3	M_3	M_1	M_2	7	4	5

deduced from this preference list. This schedule can only provide the optimal and feasible solution (Pongchairerks, 2019). Therefore, the most popular algorithm, known as G&T (Giffler & Thompson, 1960), is employed to decode a preference list of jobs to an active schedule. The Algorithm 3 explains the procedure and the corresponding notations are as follows:

(i, j): the operation of job j that need to be processed on machine i,

S: the partial schedule that contains scheduled operations,

Ω: the set of operations without predecessors,

S_{ij}: the earliest start time at which $(i, j) \in \Omega$ could be started,

P_{ij}: the processing time of (i, j),

C_{ij}: the earliest time at which $(i, j) \in \Omega$ could be completed:$C_{ij} = S_{ij} + P_{ij}$

Algorithm 3 Decoding Algorithm

```
Define Preference list; Job sequence; Processing time;
Initialize S = ∅; Ω is initialized to contains all operations without predecessors;
while True do
    Determine C* = min_{(i,j)∈Ω}{C_ij} and the machine m* on which C* could be realized;
    Identify the operations (i_o, j_o) ∈ Ω with associated starting time S_{i_o j_o} in such a way that (i_o, j_o)
    requires machine m*, and S_{i_o j_o} < C*;
    Choose (i, j) among the operation set (i_o, j_o) with higher preference;
    Add (i, j) to partial schedule set S;
    Assign S_ij as the starting time of (i, j) and C_ij = S_ij + P_ij;
    if all the operations are not assigned to S then
        Delete (i, j) from Ω and includes its immediate successor in Ω;
    else
        Stop
    end if
end while
C_j = max{C_j1, C_j2, ..., C_jm}, ∀j;
C_max = max{C_1, C_2, C_3, ..., C_n};
```

The decoding algorithm is applied on the exemplified preference list with the JSSP data set of Table 1, and thus an active schedule with makespan 22 is achieved, as depicted in Fig. 2i (each step is explained from Fig. 2a–i).

Mixed selection strategy (MSS)

The selection operator in each algorithm is responsible for the searching power of the next generation. Most of the algorithms prioritize the optimization of the fitness value, i.e. the probability of selecting an individual is proportional to the fitness value (Ren & Wang, 2012). This phenomenon leads to faster premature convergence as a result of filling the population with similar individuals (Goldberg, 2006). The MSS based on fitness value and similarity index is proposed to avoid premature convergence. Let us assume two solution orders such as $\mathbf{S}_1^G$ =[(1, 3, 2), (1, 2, 3), (2, 1, 3)] and $\mathbf{S}_2^G$ =[(2, 3, 1), (1, 2, 3), (1, 3, 2)]. The first sub-string of both solution orders are [(1, 3, 2)] and [(2, 3, 1)], which shows that only the second position has a similar operation and the remainder are different. This scenario is defined by the positional similarity degree, and the sum of the similarity degree (SPS) is 1 (0 + 1 + 0). The SPS for the rest of the two sub-strings will be 3 and 0, respectively. Therefore, the co-positional similarity degree between $\mathbf{S}_1^G$ and $\mathbf{S}_2^G$ denoted as $S^G(\mathbf{S}_1^G, \mathbf{S}_2^G)$ will be 0.44 ($\frac{1+3+0}{3*3}$), where the number of jobs (n) and the number of machines (m) are both 3. If the population comprises of NP solution orders, then the concentration value of the $\mathbf{S}_1^G$ is calculated based on Eq. 10.

$$c(\mathbf{S}_1^G) = \sum_{l=1}^{(NP-1)} \frac{S(\mathbf{S}_1^G, \mathbf{S}_l^G)}{(NP-1)} \tag{10}$$

The goodness value $G_s(\mathbf{S}_1^G)$ which consists of concentration value and the fitness value $f(\mathbf{S}_1^G)$ of the corresponding solution order is calculated using Eq. 11. The fitness value is a makespan that is calculated using G&T algorithm, as explained in Sect. 5.4. The higher the value of $G_s(\mathbf{S}_1^G)$, the better diversification and fitness value can be achieved. W is for priority between the diversity and fitness value.

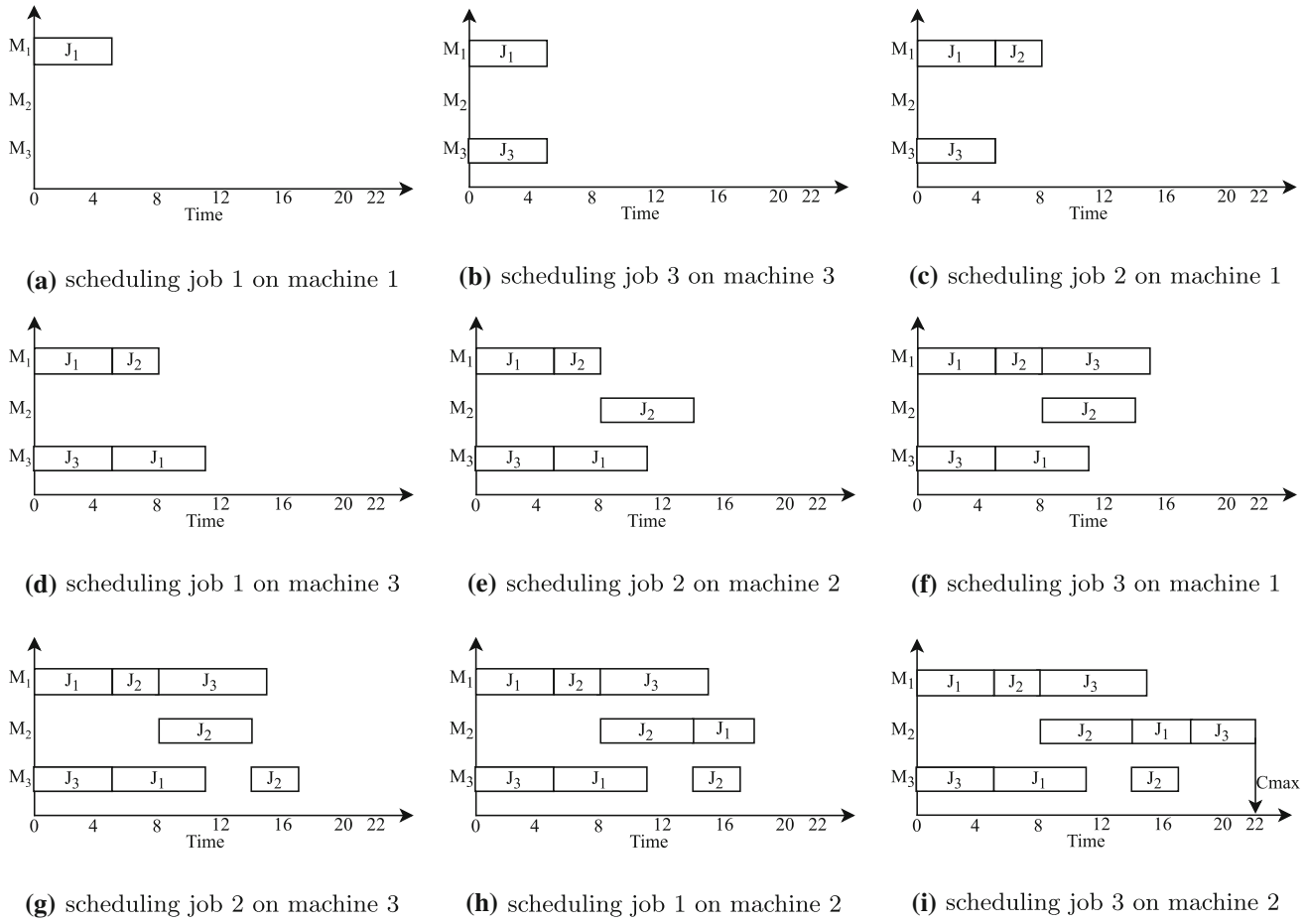
$$G_s(\mathbf{S}_1^G) = W \times \frac{max(f(\mathbf{S}^G)) - f(\mathbf{S}_1^G)}{max(f(\mathbf{S}^G)) - min(f(\mathbf{S}^G))} + (1 - W) \times (1 - c(\mathbf{S}_1^G)) \tag{11}$$

Evolution process

The strategies employed in the HEAs (i.e., sHEA, rHEA and bHEA) are discussed in this section.

Multi-operator differential evolution (MODE)

DE is conventionally represented by DE/a/b/c, in which DE is for differential evolution, a denotes the vector to be perturbed, b stands for the number of differential for the perturbation of a, and c stands for the type of crossover to be used such as bin: binomial, exp: exponential. The evolutionary process of this algorithm comprises mutation, crossover and selection. This algorithm has eight types of mutation operators presented from Eqs. 12–19, which have individual traits (Liu et al., 2020). The three mutation operators which are selected in this MODE are "DE/rand/1", "DE/best/1" and "DE/current-to-pbest/1" shown in Eqs. 12, 13, and 19, respectively. "DE/rand/1" is recognized to have the best exploration property (Zhao et al., 2016) whereas, "DE/best/1" has the best exploitation property (Wisittipanich & Kachitvichyanukul, 2012). The balance between these two characteristics in the evolutionary process is ensured by "DE/current-to-pbest/1"



(a) scheduling job 1 on machine 1 **(b)** scheduling job 3 on machine 3 **(c)** scheduling job 2 on machine 1

(d) scheduling job 1 on machine 3 **(e)** scheduling job 2 on machine 2 **(f)** scheduling job 3 on machine 1

(g) scheduling job 2 on machine 3 **(h)** scheduling job 1 on machine 2 **(i)** scheduling job 3 on machine 2

Fig. 2 Decoding to an active schedule of 3×3 JSSP

(Liu et al., 2020). These three mutation operators are combined in DE and known as MODE in this study.

$$\text{DE/rand/1:}\quad \mathbf{M}_l^G = \mathbf{x}_{r_1}^G + F \times (\mathbf{x}_{r_2}^G - \mathbf{x}_{r_3}^G) \tag{12}$$

$$\text{DE/best/1:}\quad \mathbf{M}_l^G = \mathbf{x}_{best}^G + F(\mathbf{x}_{r_1}^G - \mathbf{x}_{r_2}^G) \tag{13}$$

$$\text{DE/current/1:}\quad \mathbf{M}_l^G = \mathbf{x}_l^G + F(\mathbf{x}_{r_1}^G - \mathbf{x}_{r_2}^G) \tag{14}$$

$$\text{DE/current-to-best/1:}\quad \mathbf{M}_l^G = \mathbf{x}_l^G + F(\mathbf{x}_{best}^G - \mathbf{x}_l^G) + F(\mathbf{x}_{r_1}^G - \mathbf{x}_{r_2}^G) \tag{15}$$

$$\text{DE/rand/2:}\quad \mathbf{M}_l^G = \mathbf{x}_{r_1}^G + F(\mathbf{x}_{r_2}^G - \mathbf{x}_{r_3}^G) + F(\mathbf{x}_{r_4}^G - \mathbf{x}_{r_5}^G) \tag{16}$$

$$\text{DE/best/2:}\quad \mathbf{M}_l^G = \mathbf{x}_{best}^G + F(\mathbf{x}_{r_1}^G - \mathbf{x}_{r_2}^G) + F(\mathbf{x}_{r_3}^G - \mathbf{x}_{r_4}^G) \tag{17}$$

$$\text{DE/current-to-rand/1:}\quad \mathbf{M}_l^G = \mathbf{x}_l^G + F(\mathbf{x}_{r_1}^G - \mathbf{x}_l^G) + F(\mathbf{x}_{r_2}^G - \mathbf{x}_{r_3}^G) \tag{18}$$

$$\text{DE/current-to-pbest/1:}\quad \mathbf{M}_l^G = \mathbf{x}_l^G + F(\mathbf{x}_{best,p}^G - \mathbf{x}_l^G) + F(\mathbf{x}_{r_1}^G - \tilde{\mathbf{x}}_{r_2}^G) \tag{19}$$

It is noted that in Eq. 12, $\mathbf{x}_{r_1} \neq \mathbf{x}_{r_2} \neq \mathbf{x}_{r_3}$ are randomly selected from the current population and these operators are also different from the target vector $\mathbf{x}_l^G$. $\mathbf{x}_{best,p}^G$ in Eq. 13 is chosen from the top 15% of individuals belonged to the current population and $\tilde{\mathbf{x}}_{r_2}^G$ in Eq. 19 is taken from the union of current population and archive of $pbest$. F is scale factor, which leads to convergence speed and the population diversity (Sallam et al., 2020). Moreover, the trial vector $\mathbf{T}_l^G$ is obtained by recombining a mutant vector $\mathbf{M}_l^G$ and a target vector $\mathbf{x}_l^G$, as shown in Eq. 4. The produced trial vector is evaluated and compared with the target vector based on the fitness value. The trial vector is assigned to the target vector if the fitness value is lower in the trial vector for the minimization problem, as depicted in Eq. 6.

The performance of this algorithm greatly depends on its search operators (mutation and crossover) and control parameters. It is claimed that having a higher value of C_r is the most suitable, and the setting of F value is quite tricky for all instances (Ponsich & Coello, 2013). However, these parameters are tuned in Sect. 6.2.

Particle swarm optimization (PSO)

Although the PSO algorithm is designed for a continuous search space, this paper employs it in a discrete combinatorial optimization domain by converting the continuous-based population to discrete. The preference-based representation illustrates particles, as discussed in Sect. 5.4, while particle movement is described in terms of a swap operation. For instance, if $\mathbf{v}_{lik}^{G} = 0$ for an operation at position k on machine or sub-string i in a solution order $\mathbf{S}_{lik}^{G}$ of a swarm, the operation can move to the **pbest** or **gbest** depending on the value of c_1 and c_2 respectively. The steps of particle movements are illustrated in Algorithm 4.

Algorithm 4 Velocity update and particle movement

```
Define c1; c2;
set i ← 1;
while i ≤ m do
    set k ← 1;
    while k ≤ n do
        Choose a location k in S^G_lik;
        Identify and denote the corresponding located job by J1;
        if 0 ≤ rand(1) ≤ c1 then
            Find out the job J1 in pbest and its corresponding location denoted by k'. Denote the job at
            location k' in S^G_lik by J2;
        else if c1 < rand(1) ≤ c1 + c2 then
            Find out the job J1 in gbest and its corresponding location denoted by k'. Denote the job at
            location k' in S^G_lik by J2;
        end if
        if v_lik and v_lik' == 0 then
            Swap J1 and J2 in S^G_lik;
            Set v_lik ← 0;
        else
            Set v_lik ← (w − 1);
        end if
        Set k ← (k + 1);
    end while
    set i ← (i + 1);
end while
```

If the **pbest** of all particles becomes the same over the generations, the evolutionary process will become trapped in local optima, and therefore a diversity check mechanism was developed to keep the makespan of **pbest** solutions different (Sha & Hsu, 2006). Besides keeping the makespan of **pbest** different, it is also significant to keep the schedules different to reduce unnecessary movements. Thus, the **pbest** and **gbest** are updated in every generation considering both makespan and their corresponding solution order using Algorithm 5.

Algorithm 5 Diversity check mechanism

```
Define gbest; pbest; S^G_l;
if f(S^G_l) < f(gbest) then
    set pbest^worst ← gbest and gbest ← S^G_l;
else if f(S^G_l) > f(gbest) && f(S^G_l) < f(pbest_l) then
    Set pbest_worst ← pbest_l, and pbest_l ← S^G_l, provided that f(S^G_l) can not be equal to any f(pbest);
else if f(S^G_l) == f(pbest) then
    Set pbest ← S^G_l, provided that solution orders are different;
else if f(S^G_l) == f(gbest) then
    Set gbest ← S^G_l, provided that solution orders are different;
else if f(S^G_l) > f(gbest) && f(S^G_l) < f(pbest^worst) then
    Set pbest^worst ← S^G_l, provided that f(S^G_l) can not be equal to any f(pbest);
end if
```

Local search

The local search capability of population-based algorithms such as PSO and DE are sufficiently lower than the global search capability, and thus these algorithms suffer from premature convergence and are easily trapped in local optima (Gao et al., 2019; Lin et al., 2010). Thus, a local search is implemented to improve further the makespan value of a schedule obtained from the MODE-PSO combination. TS has been employed as it can generate solutions around the optima (Peng et al., 2015). This technique improves the solution quality iteratively and escapes the evolution of being trapped in local optima. In this study, the **pbest** solutions, which are found from the MODE and PSO, are passed onto the TS when the evolutionary process in MODE-PSO is incapable of updating the best solution obtained so far. It improves the **pbest** and **gbest**, which directly enhance the searching capability of MODE-PSO. The main steps of the implemented local search are explained in Algorithm 6.

Algorithm 6 Local search

```
Create pbest pool (5% of NP); Define parameters (see Sect. 6.2); Set Count3 ← 0; Set FES_LS ← 0;
Set l ← 1;
while l ≤ |pbest_pool| do
    Assign S_l ← pbest_l as a candidate solution;
    while (Count3 %TabuTermination ≠ 0) do
        Find a critical path of S_l and create neighbour solutions using Figure 3;
        Evaluate neighbour solutions; FES_LS ← (FES_LS + nNeighbourSolution);
        Choose the best admissible candidate based on the Tabu restrictions and aspiration criteria and
        assign as S_l; Update the Tabu list;
        if any(f(S_l) < [f(pbest_l); f(gbest); f(pbest^worst)]) then
            Update pbest_l, gbest or pbest^worst using Algorithm 5;
            if any([f(pbest_l); f(gbest); f(pbest^worst)]<previous([f(pbest_l); f(gbest); f(pbest^worst)]))
            then
                Stop;
            end if
        end if
        if f(gbest) < f(S_l) then
            Count3 ← (Count3 + 1);
        end if
    end while
    Set l ← (l + 1);
end while
```

Neighbourhood structure A small perturbation in the **pbest** can produce a set of solutions known as neighbour. In this study, these neighbour solutions are generated by implementing the $N7$ neighbourhood structure (Zhang et al., 2007), depicted in Fig. 3. In this approach, the randomly selected critical path, as a solution that could have multiple critical paths, is divided into several blocks, making moves in each block for new neighbours.

Tabu list Tabu list helps the algorithm to resist visiting the same solution recurrently, avoiding becoming trapped in local optima. Instead of storing solution attributes such as makespan, the sequence of operations and their corresponding position in a machine are stored in a Tabu list for a certain number of iterations, called Tabu tenure. The Tabu tenure dynamically adjusts based on the problem size of JSSP instances (see Table 3). The aspiration criterion is allowed to accept the best solution obtained so far even after the solution is in the Tabu list.

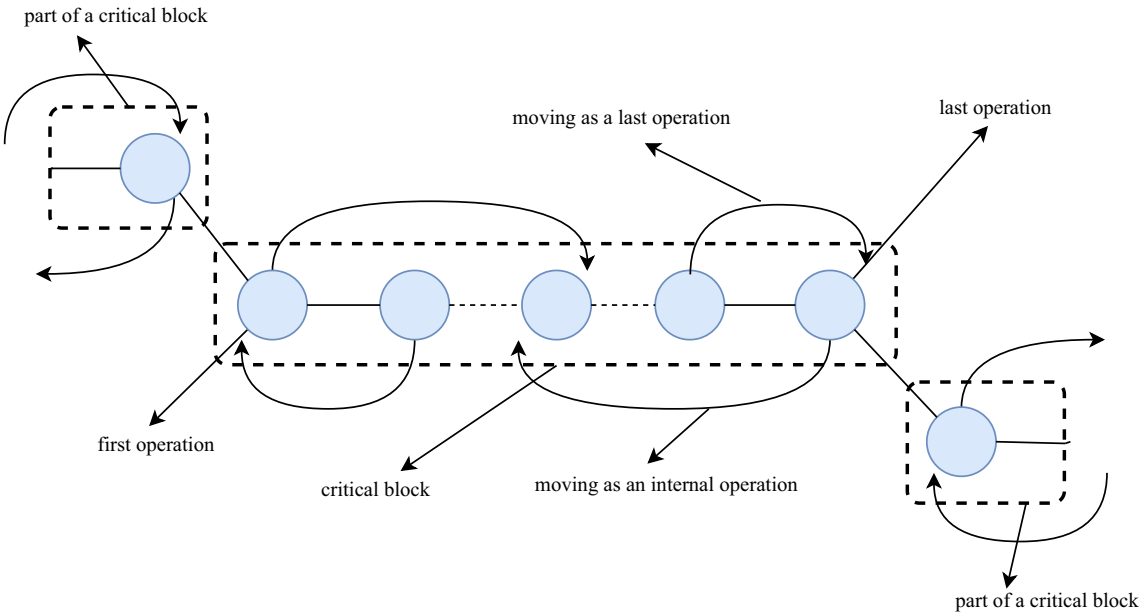


Fig. 3 The outline of the N7 neighbourhood

Performance-driven switching mechanism (PDSM)

As previously mentioned, in the evolution process, three evolution strategies such as MODE, PSO and local search (LS) are used in a single algorithmic framework. So, the proposed algorithms dynamically choose the most appropriate strategy at a different level of evolution depending on the performance of the evolution strategies, named PDSM. It offers two benefits:(1) if any algorithm performs adequately over the generations, no other algorithms evolve to solve the problem. It will increase the flexibility in the recently emerged multi-method approaches to solve complex problems in which a single method is incapable; and (2) Over the evolution process, an algorithm may perform poorly, and it is inappropriate to carry out the search process. This approach overcomes the drawback. Equation 20 controls the switching decision. If any algorithm performs poorly, $Count_{alg|\{alg=1,2.3\}}$ will reach quickly to $PreMax$ or Tabu termination, and the population switches to another search strategy.

Experimental design and result analysis

The section illustrates the computing environment, parameter analysis to achieve the best performance of the proposed algorithms, and detailed result analysis.

Experimental setting

The computing environment used in this research is Intel(R) Core (TM) i7-3770 CPU @3.40GHz with RAM 32.0 GB in Windows 10, and the proposed algorithms (i.e., sHEA, rHEA and bHEA) are coded in MATLAB R2020b. The feasibility and effectiveness of these algorithms are investigated through a set of comparative experiments with the TS, DE and PSO and against the 26 popular algorithms by solving 130 standard instances of the JSSP, including rectangular problems and the most difficult square problems (Cruz-Chávez et al., 2019). The selected instances include: 3 from Fisher and Thompson (FT) (Fisher, 1963); 40 from Lawrence (LA) (Lawrence, 1984); 3 from Applegate and Cook (ORB) (Applegate & Cook, 1991); 4 from Adams,

$$Count_{alg|\{alg=1,2.3\}} = \begin{cases} Count_{alg|\{alg=1,2.3\}} + 1 & \text{if } f(\mathbf{gbest}^{\mathbf{G}}) < f(\mathbf{x}_l^G), \forall l = 1, 2, 3, \ldots, NP \\ Count_{alg|\{alg=1,2.3\}} & \text{otherwise} \end{cases} \tag{20}$$

Balas and Zawack (ABZ) (Adams et al., 1988); 80 from Taillard (TA) (Taillard, 1993), considering problems with varied sizes. Each instance is executed 30 times independently, and the corresponding results are recorded.

Parameter setting

Parameter calibration is essential to achieve the best performance of an algorithm. This section discusses the parameter tuning of the developed algorithms: the sHEA, rHEA and bHEA. The algorithms rHEA and sHEA are distinct only by population switching decision, as illustrated in Sect. 5.2, which does not imply parameter calibration, and therefore, parameter tuning is performed similarly in each algorithm. However, bHEA, unlike rHEA and sHEA, is designed with a bi-population structure where the initial population is divided into two sub-populations and executed in MODE and PSO independently, as described in Sect. 5.1. Thus, the parameters NP, $MaxGen(=\frac{MAX_{FES}}{NP})$, and $PreMax$ may have a different impact on the bi-population structure of bHEA compared to rHEA and sHEA, and therefore, calibration is performed separately for bHEA.

The parameters of MSS, PSO and MODE, which are designed in a similar way for sHEA, rHEA and bHEA, should be appropriately tuned in the first step, which is conducted by choosing a complex square-sized instance LA40. The chosen parameters with their levels are depicted in Table 2. Although the parametric analysis can be performed using trial-and-error technique (Goli et al., 2021), this study adopted the orthogonal array, $L_{27}3^5$, to design the experimental runs as it reduces the computational complexity through lessening the number of experiments (Rahman et al., 2020). However, each treatment is executed for 30 times, keeping the NP, $MaxGen$ and $PreMax$ constant. These parameters significantly impact achieving preferred makespan while the algorithm solves instances. The analyzed results are depicted in Table 2 as a response for means. Figure 4a also shows that the makespan decreases with increasing C_r, and therefore, further conducted experiments shows average makespan of 1260.4 and 1260.8 for C_r value $rand_{0.81-1.0}$ and $rand_{0.85-1.0}$, respectively. Thus, the statistical inference on the optimal setting of these parameters is $W = 0.5$, $w = 1$, $c_1 = 0.5$, $c_2 = 0.5$, $C_r = rand_{0.81-1.00}$, $F = rand_{0.91-1.30}$. The values $rand_{0.81-1.00}$ and $rand_{0.91-1.30}$ indicate that the parameters adopt randomly generated values in the given range following a uniform distribution.

The parameters of the integrated frameworks (i.e., rHEA and sHEA) are also tuned. The five significant parameters of the rHEA and sHEA are set into three levels, as reported in Table 3, and designed using the orthogonal array, $L_{27}3^5$. Each of the 27 treatments is run 30 times, and the average makespan of LA40 is recorded. The analysed results, including the delta value and rank of parameters, are reported in the above table. Following the table, $MaxGen$ is ranked first, whereas, Tabu termination (i.e., local search termination criteria) is ranked the second most crucial factor. The $PreMax$ (i.e., terminating PSO and MODE at a defined level if the best value is not updated) is ranked third. Tabu tenure and NP are the less significant parameters ranked fourth and fifth, respectively. However, more analysis is needed as makespan improves with increasing NP and $MaxGen$ and with decreasing $PreMax$ as depicted in Fig. 4b, and these three parameters significantly influence the computational expenses. However, Tabu tenure is excluded to lessen the complexity as it is a less significant parameter. In addition, this analysis offers to set optimal parameters for bHEA, as illustrated previously.

The further impact of NP, $MaxGen$ and $PreMax$ on the performance of rHEA and sHEA are determined through sensitivity analysis. This experiment uses both rectangular- and square-sized instances (e.g., $LA17$, $LA30$ and $LA40$). For each selected instance, parameters NP, $MaxGen$ and $PreMax$ are varied with possible values $NP = \{50, 75, 100, 125\}$; $MaxGen = \{1000, 1500, 2000, 2500\}$, and $PreMax = \{20, 16, 12, 10\}$. For each parameter combination, 10 independent runs are performed, and the relative percentage error (RPE) of each algorithm's best solution (EBS) found and RPE of the mean value of the makespan f_{mean} are calculated based on the reference makespan known as the best known solution (BKS), as depicted in Fig. 5a. The value of RPE and mean RPE are calculated by Eq. 21. $RPE = 0$ indicates that the instance is solved optimally. From the Fig. 5a, the algorithms sHEA and rHEA show almost consistent performance from parameter combinations 3 to 7 for each of the instances, and therefore, NP, $MaxGen$ and $PreMax$ are set to 100, 1000 and 20, respectively. These settings minimize the number of fitness evaluation meant to less computational expenses. For bHEA, a similar experiment is performed as depicted in Fig. 5b and the performance for each experimental setup and for each instance shows that parameter combinations 3 to 8 offer relatively consistent and better performance. Therefore, the parameter combination 3, which is similar to sHEA and rHEA, is recommended for bHEA for fair comparison. According to this parametric analysis, the optimal parameter setting for sHEA, rHEA and bHEA are as follows: $NP = 100$; $MaxGen = 1000$; $PreMax = 20$, Tabu termination=15, and Tabu tenure= $L_{min} = [10 + \frac{m}{n}]$ to $L_{max} = 1.4L_{min}$.

Comparison with constituent algorithms

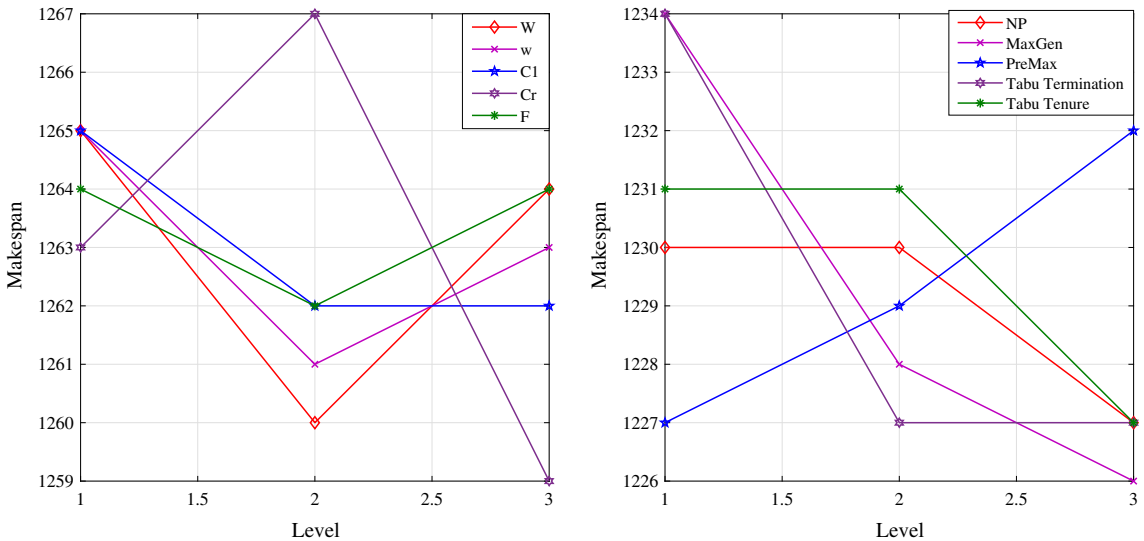
This section aims to validate the hypothesis that integrating multiple algorithms' capabilities into a single algorithmic framework can find the best possible solutions. Thus, several experiments are conducted to verify the hypothesis, implementing the sHEA, rHEA, bHEA, and their con-

Table 2 The parameters tuning of the MSS-MODE-PSO

Parameters settings					Response for means				
Parameters		Level			Level			Delta	Rank
		1	2	3	1	2	3		
MSS	W	0.2	0.5	0.8	1265	1260	1264	5	2
PSO	w	0	1	2	1265	1261	1263	4	3
	$c_1(c_2)$	0.2 (0.8)	0.5 (0.5)	0.8 (0.2)	1265	1262	1262	4	4
MODE	C_r	$rand_{0.30-0.50}$	$rand_{0.51-0.80}$	$rand_{0.81-1.00}$	1263	1267	1259	8	1
	F	$rand_{0.20-0.90}$	$rand_{0.91-1.30}$	$rand_{1.31-2.00}$	1264	1262	1264	2	5

Table 3 Parameters tuning of rHEA and sHEA

Parameter settings				Response for means				
Parameters	Level			Level			Delta	Rank
	1	2	3	1	2	3		
NP	50	75	100	1230	1230	1227	3	5
MaxGen	500	750	1000	1234	1228	1226	8	1
PreMax	20	30	40	1227	1229	1232	5	3
Tabu Termination	5	10	15	1234	1227	1227	7	2
Tabu Tenure	$L_{min} = 5 + \frac{n}{m}$	$L_{min} = 7 + \frac{n}{m}$	$L_{min} = 10 + \frac{n}{m}$	1231	1231	1227	4	4
	$L_{max} = 0.4L_{min}$	$L_{max} = 0.4L_{min}$	$L_{max} = 0.4L_{min}$					

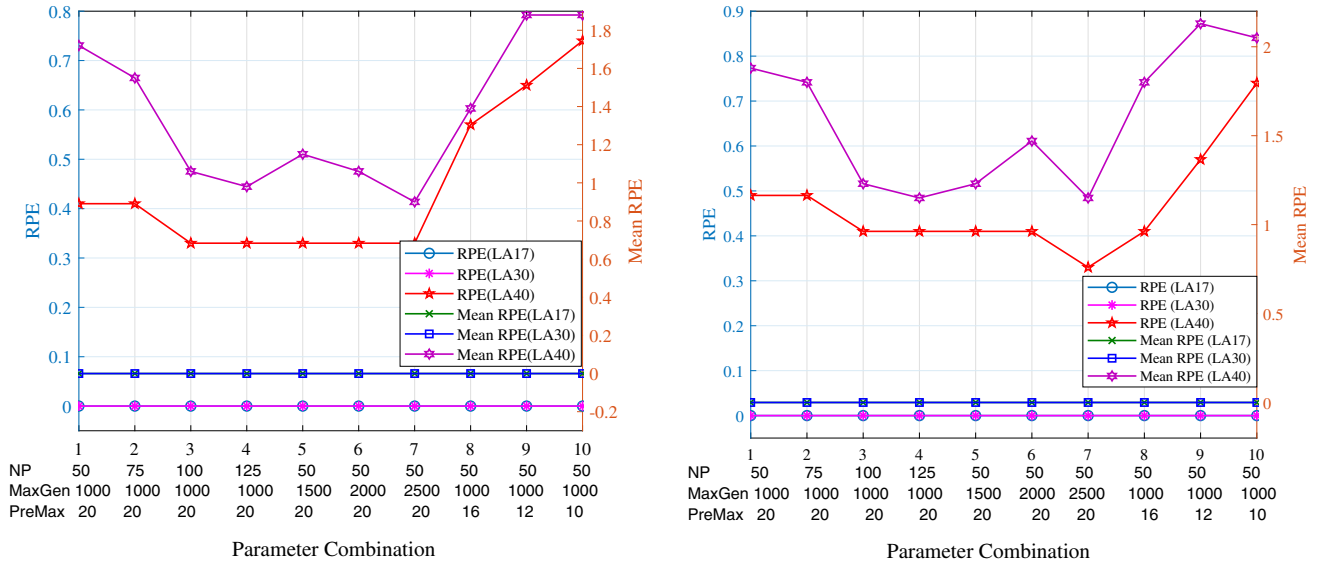


(a) MSS-MODE-PSO **(b)** rHEA and sHEA

Fig. 4 Main effects plot for means

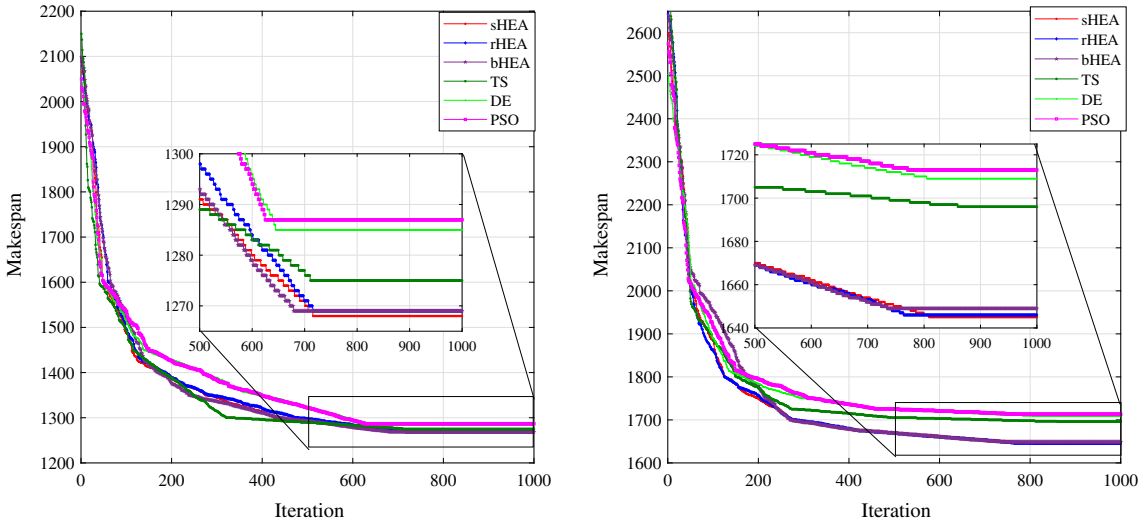
stituent algorithms, such as PSO, DE and TS, to solve the computationally intractable NP-hard JSSPs. The statistical convergence curves of the two varied-sized instances (LA36 (15 × 15) and TA20 (20 × 20)) are depicted in Fig. 6.

The convergence of the original PSO and DE is faster at the very beginning for both LA36 and TA21 instances, as illustrated in Fig. 6a and b, respectively. However, the solution is not improved later due to the greedy nature of DE and the poor population quality. TS inspired local search has experienced a fast convergence rate over the iterations and found near-optimal solutions for both instances by avoiding becoming trapped in local optima. However, the single point searching process is computationally expensive. Thus, TS is employed in hybrid MODE-PSO to improve the local



(a) rHEA and sHEA **(b)** bHEA

Fig. 5 Optimal parameter setting through sensitivity analysis



(a) LA36 (15×15) **(b)** TA21(20×20)

Fig. 6 Convergence curve of the six implemented algorithms

search capability. The integrated approaches (sHEA, rHEA and bHEA), where PSO, MODE and TS are merged with the help of the switching strategy, shows that the searching power of the evolutionary process is improved substantially in terms of global search capability at the early stage and local search capability at the later stage, leading to higher solution quality with mature convergence.

Performance of the proposed HEAs (i.e., sHEA, rHEA and rHEA) against original TS, DE and PSO is further evaluated by solving 12 standard JSSP instances. The recorded results are reported in Table 4, where the best performance of each instance is marked as bold. The Table includes instances name, their size, BKS, EBS, RPE, BKS of each algorithm found (BKS found) and the ratio of the number of BKS to the number of benchmark instances solved (NIS) (Ratio(%)). RPE(%) and Ratio(%) are calculated by Eqs. 21 and 22, respectively.

The Table shows that all the algorithms implemented have solved 12 instances with different sizes from 6 × 6 JSSP to 20×20 JSSP. Out of the 12 instances solved, TS, DE and PSO obtain 8, 7 and 8 BKS, respectively, whereas sHEA, rHEA and bHEA obtain 10, 10 and 8 BKS, respectively. Although

Table 4 Comparison among implemented algorithms (PSO, DE, TS, and HEAs) for JSSP

Instance	Size (m×n)	BKS	sHEA		rHEA		bHEA		TS		DE		PSO	
			EBS	RPE	EBS	RPE	EBS	RPE	EBS	RPE	EBS	RPE	EBS	RPE
FT06	6×6	55	55	**0**	55	**0**	55	**0**	55	0.00	55	0.00	55	0.00
FT10	10×10	930	930	**0**	930	**0**	930	**0**	930	0.00	930	0.00	930	0.00
FT20	20×5	1165	1165	**0**	1165	**0**	1165	**0**	1165	0.00	1165	0.00	1165	0.00
LA01	10×5	666	666	**0**	666	**0**	666	**0**	666	0.00	666	0.00	666	0.00
LA06	15×5	926	926	**0**	926	**0**	926	**0**	926	0.00	926	0.00	926	0.00
LA11	20×5	1222	1222	**0**	1222	**0**	1222	**0**	1222	0.00	1222	0.00	1222	0.00
LA16	10×10	945	945	**0**	945	**0**	945	**0**	945	0.00	945	0.00	945	0.00
LA21	15×10	1046	1046	**0**	1046	**0**	1046	**0**	1046	0.00	1052	0.57	1046	0.00
LA36	15×15	1268	1268	**0**	1269	0.08	1269	0.08	1275	0.55	1285	1.34	1287	1.50
TA01	15×15	1231	1231	**0**	1231	**0**	1235	0.32	1248	1.38	1253	1.79	1259	2.27
TA11	20×15	1357	1359	0.15	1357	**0**	1361	0.29	1374	1.25	1386	2.14	1374	1.25
TA21	20×20	1642	1645	**0.18**	1646	0.24	1649	0.43	1696	3.29	1709	4.08	1713	4.32
BKS found			10		10		8		8		7		8	
Ratio (%)			83.33		83.33		66.67		66.67		58.33		66.67	

Table 5 Improvement comparison among implemented algorithms (PSO, DE, TS, and HEAs)

Algorithm	NIS	OA(%)	sHEA		rHEA		bHEA	
			aRPE(%)	RPI (%)	aRPE(%)	RPI (%)	aRPE(%)	RPI (%)
PSO	12	0.78	0.03	96.47	0.03	96.55	0.09	87.97
DE	12	0.83	0.03	96.67	0.03	96.75	0.09	88.66
TS	12	0.54	0.03	94.90	0.03	95.02	0.09	82.63

the implemented algorithms show similar performance for small-sized instances, a significant improvement of hybrid approaches is noticed for large-sized problems. Moreover, the improvement comparison between the HEAs and the other algorithms (i.e., original TS, DE and PSO) are given in Table 5 that presents NIS, average RPE (aRPE(%)) of both other algorithms (OA(%)) and proposed algorithms calculated by Equation 23, and relative performance improvement (RPI(%)) calculated by Eq. 24. If RPI(%) is negative, the proposed algorithm can not outperform the other algorithms. Following RPI, bHEA dominates significantly compared to PSO by 87.97%, DE by 88.66% and TS by 82.63%, whereas rHEA shows superior performance to PSO by 96.55%, DE by 96.75% and TS by 95.02%. The sHEA dominates PSO, DE and TS by 96.47%, 96.67% and 94.90%, respectively. Thus, the considered hypothesis is true, i.e., integrating multiple algorithms' capabilities into a single algorithmic framework can find the best possible solutions.

$$RPE(\%) = \frac{EBS - BKS}{BKS} \times 100 \tag{21}$$

$$Ratio(\%) = \frac{BKS\ found}{NIS} \times 100 \tag{22}$$

$$aRPE(\%) = \frac{\sum RPE}{NIS} \tag{23}$$

$$RPI(\%) = \frac{OA - aRPE}{OA} \times 100 \tag{24}$$

Comparison with well-know algorithms

The section assesses the effectiveness of the sHEA, rHEA and bHEA against existing algorithms in the literature by solving small-sized and large-sized JSSP instances. First, 13 popular algorithms in the literature are considered. These algorithms include widely accepted variants of DE, such as DE with strategy switching (DESS) (Wisittipanich & Kachitvichyanukul, 2012), hybrid DE and estimation of distribution algorithm with neighbourhood search known as (NS-HDE/EDA) (Zhao et al., 2016), and whale optimization algorithm enhanced with lévy flight and DE (WOA-LFDE) (Liu et al., 2020). In addition to DE variants, some other considered algorithms are filter-and-fan approach (F&F) (Rego & Duarte, 2009), TLBO (Baykasoğlu et al., 2014), chemotaxis-enhanced bacterial foraging algorithm (CEBFO) (Zhao et al., 2015), parallel artificial bee colony algorithm (pABC) (Asadzadeh, 2016), GWO (Jiang & Zhang, 2018) and parallel bat algorithm (PBA) (Dao et al., 2018). The developed algorithms are further evaluated by comparing with the variants of GA, such as memetic algorithm (MA) (Gao et al., 2011), a local search GA (aLSGA) (Asadzadeh,

Table 6 Comparison of the 13 algorithms with HEAs for JSSP

Instance	Size ($n \times m$)	BKS	F&F		MA		DESS		TLBO		MGA	
			$\overline{\text{EBS}}$	$\overline{\text{RPE}}$	$\overline{\text{EBS}}$	$\overline{\text{RPE}}$	$\overline{\text{EBS}}$	$\overline{\text{RPE}}$	$\overline{\text{EBS}}$	$\overline{\text{RPE}}$	$\overline{\text{EBS}}$	$\overline{\text{RPE}}$
FT06	6×6	55	55	0	55	0	55	0	55	0	55	0
FT10	10×10	930	930	0	930	0	949	2.04	938	0.86	930	0
FT20	20×5	1165	1165	0	1165	0	1204	3.35	1165	0	1165	0
LA01	10×5	666	666	0	666	0	666	0	666	0	666	0
LA02	10×5	655	655	0	655	0	655	0	655	0	655	0
LA03	10×5	597	597	0	597	0	597	0	597	0	597	0
LA04	10×5	590	590	0	590	0	590	0	607	2.88	590	0
LA05	10×5	593	593	0	593	0	593	0	593	0	593	0
LA06	15×5	926	926	0	926	0	926	0	926	0	926	0
LA07	15×5	890	890	0	890	0	890	0	890	0	890	0
LA08	15×5	863	863	0	863	0	863	0	864	0.12	863	0
LA09	15×5	951	951	0	951	0	951	0	951	0	951	0
LA10	15×5	958	958	0	958	0	958	0	958	0	958	0
LA11	20×5	1222	1222	0	1222	0	1222	0	1222	0	1222	0
LA12	20×5	1039	1039	0	1039	0	1039	0	∼	∼	1039	0
LA13	20×5	1150	1150	0	1150	0	1150	0	∼	∼	1150	0
LA14	20×5	1292	1292	0	1292	0	1292	0	∼	∼	1292	0
LA15	20×5	1207	1207	0	1207	0	1207	0	∼	∼	1207	0
LA16	10×10	945	947	0.21	945	0	945	0	946	0.11	945	0
LA17	10×10	784	784	0	784	0	784	0	∼	∼	784	0
LA18	10×10	848	848	0	848	0	848	0	∼	∼	848	0
LA19	10×10	842	846	0.48	842	0	851	1.07	∼	∼	842	0
LA20	10×10	902	907	0.55	902	0	907	0.55	∼	∼	902	0
LA21	15×10	1046	1052	0.57	1055	0.86	1083	3.54	1091	4.30	1053	0.67

Instance	CEBFO		aLSGA		NS-HDE/EDA			NIMGA			pABC	
	$\overline{\text{EBS}}$	$\overline{\text{RPE}}$	$\overline{\text{EBS}}$	$\overline{\text{RPE}}$	$\overline{\text{EBS}}$	$\overline{\text{RPE}}$	CT	$\overline{\text{EBS}}$	$\overline{\text{RPE}}$	CT	$\overline{\text{EBS}}$	$\overline{\text{RPE}}$
FT06	55	0	55	0	55	0	2.31	55	0	1	55	0
FT10	937	0.75	930	0	937	0.75	328.60	930	0	76	930	0
FT20	1171	0.52	1165	0	1178	1.12	418.03	1173	0.69	84.64	1165	0
LA01	666	0	666	0	666	0	2.66	666	0	1	666	0
LA02	655	0	655	0	655	0	22.09	655	0	13	655	0
LA03	597	0	606	1.51	597	0	1835.00	597	0	15	597	0
LA04	590	0	593	0.51	590	0	24.08	590	0	19	590	0
LA05	593	0	593	0	593	0	0.12	593	0	2	593	0
LA06	926	0	926	0	926	0	0.39	926	0	2	926	0
LA07	890	0	890	0	890	0	5.08	890	0	2	890	0
LA08	863	0	863	0	863	0	1126.00	863	0	2	863	0
LA09	951	0	951	0	951	0	4.49	951	0	2	951	0
LA10	958	0	958	0	958	0	0.15	958	0	1	958	0
LA11	1222	0	1222	0	1222	0	1.63	1222	0	2	1222	0
LA12	1039	0	1039	0	1039	0	2.11	1039	0	2	1039	0
LA13	1150	0	1150	0	1150	0	8.66	1150	0	2	1150	0
LA14	1292	0	1292	0	1292	0	0.07	1292	0	2	1292	0
LA15	1207	0	1207	0	1207	0	13.87	1207	0	3	1207	0
LA16	945	0	946	0.11	956	1.16	90.87	946	0.11	50.07	945	0
LA17	785	0.13	784	0	784	0	81.59	784	0	30	784	0

Table 6 continued

Instance	CEBFO		aLSGA		NS-HDE/EDA			NIMGA			pABC	
	EBS	RPE	EBS	RPE	EBS	RPE	CT	EBS	RPE	CT	EBS	RPE
LA18	848	0	848	0	855	0.83	102.82	848	0	33	848	0
LA19	844	0.24	852	1.19	852	1.19	103.20	842	0	14	842	0
LA20	907	0.55	907	0.55	907	0.55	92.16	907	0.55	73.69	907	0.55
LA21	∼	∼	1068	2.10	1058	1.15	504.17	1058	1.15	100.33	1046	0

Instance	GWO			PBA		WOA-LFDE			sHEA			
	EBS	RPE	CT	EBS	RPE	EBS	RPE	CT	EBS	Avg.	RPE	CT
FT06	55	0	6.60	55	0	55	0	∼	**55**	55.00	0	2.85
FT10	940	1.08	62.20	930	0	930	0	∼	**930**	932.90	0	61.23
FT20	1178	1.12	76.10	1165	0	1165	0	∼	**1165**	1165.00	0	60.25
LA01	666	0	13.80	666	0	666	0	∼	**666**	666.00	0	7.12
LA02	655	0	14.50	655	0	655	0	∼	**655**	655.00	0	14.25
LA03	597	0	14.30	597	0	597	0	∼	**597**	597.00	0	8.20
LA04	590	0	12.90	590	0	590	0	∼	**590**	590.00	0	15.23
LA05	593	0	13.60	593	0	593	0	∼	**593**	593.00	0	14.23
LA06	926	0	36.10	926	0	926	0	∼	**926**	926.00	0	12.25
LA07	890	0	34.90	890	0	890	0	∼	**890**	890.00	0	18.01
LA08	863	0	38.10	863	0	863	0	∼	**863**	863.00	0	6.29
LA09	951	0	35.90	951	0	951	0	∼	**951**	951.00	0	7.36
LA10	958	0	34.00	958	0	958	0	∼	**958**	958.00	0	9.65
LA11	1222	0	71.30	1222	0	1222	0	∼	**1222**	1222.00	0	4.56
LA12	1039	0	72.00	1039	0	1039	0	∼	**1039**	1039.00	0	7.56
LA13	1150	0	71.40	1150	0	1150	0	∼	**1150**	1150.00	0	3.99
LA14	1292	0	67.50	1292	0	1292	0	∼	**1292**	1292.00	0	4.75
LA15	1207	0	74.60	1207	0	1207	0	∼	**1207**	1207.00	0	7.69
LA16	956	1.16	61.10	945	0	945	0	∼	**945**	945.60	0	69.21
LA17	790	0.77	60.30	784	0	784	0	∼	**784**	784.00	0	68.29
LA18	859	1.30	58.90	848	0	848	0	∼	**848**	848.00	0	65.09
LA19	845	0.36	61.20	842	0	842	0	∼	**842**	842.00	0	75.51
LA20	937	3.88	60.80	902	0	902	0	∼	**902**	909.97	0	71.91
LA21	1090	4.21	153.40	1046	0	1046	0	∼	**1046**	1066.43	0	110.30

Instance	rHEA				bHEA			
	EBS	Avg.	RPE	CT	EBS	Avg.	RPE	CT
FT06	**55**	55.00	0	3.12	**55**	55.00	0	2.35
FT10	**930**	942.67	0	68.25	**930**	944.43	0	78.20
FT20	**1165**	1165.00	0	52.31	**1165**	1165.00	0	55.23
LA01	**666**	666.00	0	6.25	**666**	666.00	0	5.25
LA02	**655**	655.00	0	13.85	**655**	656.10	0	9.25
LA03	**597**	597.00	0	12.29	**597**	597.47	0	12.28
LA04	**590**	590.00	0	17.12	**590**	590.00	0	13.15
LA05	**593**	593.00	0	14.65	**593**	593.00	0	12.85
LA06	**926**	926.00	0	10.61	**926**	926.00	0	8.65
LA07	**890**	890.13	0	8.85	**890**	890.53	0	9.26

Table 6 continued

Instance	rHEA				bHEA			
	EBS	Avg.	RPE	CT	EBS	Avg.	RPE	CT
LA08	**863**	863.00	0	7.98	**863**	863.00	0	3.29
LA09	**951**	951.00	0	8.95	**951**	951.00	0	6.59
LA10	**958**	958.00	0	9.86	**958**	958.00	0	7.29
LA11	**1222**	1222.00	0	6.23	**1222**	1222.00	0	5.26
LA12	**1039**	1039.00	0	4.58	**1039**	1039.00	0	4.98
LA13	**1150**	1150.00	0	4.97	**1150**	1150.00	0	7.62
LA14	**1292**	1292.00	0	5.25	**1292**	1292.00	0	4.91
LA15	**1207**	1207.00	0	4.78	**1207**	1209.10	0	4.37
LA16	**945**	958.30	0	82.31	**945**	969.33	0	78.26
LA17	**784**	784.77	0	81.61	**784**	784.80	0	69.89
LA18	**848**	848.00	0	74.38	**848**	848.87	0	72.25
LA19	**842**	842.27	0	79.20	**842**	844.60	0	64.36
LA20	**902**	908.80	0	76.12	**902**	910.00	0	72.26
LA21	**1046**	1067.77	0	125.12	**1046**	1081.63	0	102.85

Instance	Size ($n \times m$)	BKS	F&F		MA		DESS		TLBO		MGA	
			EBS	RPE	EBS	RPE	EBS	RPE	EBS	RPE	EBS	RPE
LA22	15×10	927	927	0	927	0	956	3.13	∼	∼	927	0
LA23	15×10	1032	1032	0	1032	0	1032	0	∼	∼	1032	0
LA24	15×10	935	941	0.64	940	0.53	978	4.60	∼	∼	941	0.64
LA25	15×10	977	982	0.51	984	0.72	1016	3.99	∼	∼	978	0.10
LA26	20×10	1218	1218	0	1218	0	1267	4.02	∼	∼	1218	0
LA27	20×10	1235	1242	0.57	1261	2.11	1306	5.75	1256	1.70	1262	2.19
LA28	20×10	1216	1225	0.74	1216	0.00	1273	4.69	∼	∼	1216	0
LA29	20×10	1152	1176	2.08	1190	3.30	1249	8.42	∼	∼	1163	0.95
LA30	20×10	1355	1355	0	1355	0	1386	2.29	∼	∼	1355	0
LA31	30×10	1784	1784	0	1784	0	1784	0	1784	0	1784	0
LA32	30×10	1850	1850	0	1850	0	1850	0	∼	∼	1850	0
LA33	30×10	1719	1719	0	1719	0	1719	0	∼	∼	1719	0
LA34	30×10	1721	1721	0	1721	0	1749	1.63	∼	∼	1721	0
LA35	30×10	1888	1888	0	1888	0	1888	0	∼	∼	1888	0
LA36	15×15	1268	1281	1.03	1281	1.03	1313	3.55	1332	5.05	1283	1.18
LA37	15×15	1397	1418	1.50	1431	2.43	1451	3.87	∼	∼	1424	1.93
LA38	15×15	1196	1213	1.42	1216	1.67	1284	7.36	∼	∼	1218	1.84
LA39	15×15	1233	1250	1.38	1241	0.65	1283	4.06	∼	∼	1251	1.46
LA40	15×15	1222	1228	0.49	1233	0.90	1289	5.48	1241	1.55	1233	0.90
ORB01	10×10	1059	∼	∼	∼	∼	∼	∼	∼	∼	1064	0.47
ORB05	10×10	887	∼	∼	∼	∼	∼	∼	∼	∼	889	0.23
ORB10	10×10	944	∼	∼	∼	∼	∼	∼	∼	∼	944	0
ABZ5	10×10	1234	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼
ABZ6	10×10	943	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼

Table 6 continued

Instance	Size ($n \times m$)	BKS	F&F		MA		DESS		TLBO		MGA	
			EBS	RPE	EBS	RPE	EBS	RPE	EBS	RPE	EBS	RPE
ABZ8	20×15	665	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼
ABZ9	20×15	679	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼
NIS			43		43		43		20		46	
BKS found			29		33		24		12		34	
Ratio (%)			67.44		76.74		55.81		60.00		73.91	

Instance	CEBFO		aLSGA		NS-HDE/EDA			NIMGA			pABC	
	EBS	RPE	EBS	RPE	EBS	RPE	CT	EBS	RPE	CT	EBS	RPE
LA22	∼	∼	956	3.13	952	2.70	640.03	937	1.08	101.42	927	0
LA23	∼	∼	1032	0	1038	0.58	292.30	1032	0	25	1032	0
LA24	∼	∼	966	3.32	973	4.06	662.72	947	1.28	72.34	935	0
LA25	∼	∼	1002	2.56	1000	2.35	701.42	992	1.54	99.19	977	0
LA26	∼	∼	1223	0.41	1229	0.90	894.96	1218	0	119	1218	0
LA27	∼	∼	1281	3.72	1287	4.21	972.42	1269	2.75	131.83	1235	0
LA28	∼	∼	1245	2.38	1275	4.85	975.70	1247	2.55	123.18	1216	0
LA29	∼	∼	1230	6.77	1220	5.90	998.91	1241	7.73	124.61	1157	0.43
LA30	∼	∼	1355	0	1371	1.18	956.76	1355	0	84	1355	0
LA31	∼	∼	1784	0	1784	0	296.60	1784	0	6	1784	0
LA32	∼	∼	1850	0	1850	0	671.69	1850	0	10	1850	0
LA33	∼	∼	1719	0	1719	0	184.71	1719	0	10	1719	0
LA34	∼	∼	1721	0	1721	0	917.05	1721	0	55	1721	0
LA35	∼	∼	1888	0	1888	0	604.61	1888	0	27	1888	0
LA36	∼	∼	∼	∼	1315	3.71	697.07	1293	1.97	110.98	∼	∼
LA37	∼	∼	∼	∼	1465	4.87	654.04	1439	3.01	138.27	∼	∼
LA38	∼	∼	∼	∼	1244	4.01	628.99	1222	2.17	138.48	∼	∼
LA39	∼	∼	∼	∼	1291	4.70	749.59	1251	1.46	137.20	∼	∼
LA40	∼	∼	∼	∼	1277	4.50	720.32	1246	1.96	138.78	∼	∼
ORB01	∼	∼	1092	3.12	∼	∼	∼	1059	1.96	∼	∼	∼
ORB05	∼	∼	901	1.58	∼	∼	∼	893	1.96	∼	∼	∼
ORB10	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼
ABZ5	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼
ABZ6	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼
ABZ8	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼
ABZ9	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼
NIS	23		40		43			45			38	
BKS found	18		25		22			28			36	
Ratio (%)	78.28		62.50		51.16			62.22			94.74	

Instance	GWO			PBA		WOA-LFDE			sHEA			
	EBS	RPE	CT	EBS	RPE	EBS	RPE	CT	EBS	Avg.	RPE	CT
LA22	970	4.64	154.30	933	0.65	927	0	∼	**927**	941.07	0	106.23
LA23	1032	0	150.90	1032	0	1032	0	∼	**1032**	1032.00	0	89.62
LA24	982	5.03	155.80	941	0.64	935	0	∼	**935**	953.43	0	106.98
LA25	1008	3.17	160.50	977	0	977	0	∼	**977**	991.97	0	115.26
LA26	1239	1.72	324.50	1218	0	1218	0	∼	**1218**	1218.00	0	135.23
LA27	1290	4.45	314.80	1247	0.97	1235	0	∼	**1235**	1269.33	0	162.32

Table 6 continued

Instance	GWO			PBA		WOA-LFDE			sHEA			
	EBS	RPE	CT	EBS	RPE	EBS	RPE	CT	EBS	Avg.	RPE	CT
LA28	1263	3.87	325.00	1216	0	1216	0	∼	**1216**	1216.00	0	125.32
LA29	1244	7.99	313.20	1179	2.34	1161	0.78	∼	**1152**	1209.30	0	155.32
LA30	1355	0	316.40	1355	0	1355	0	∼	**1355**	1355.00	0	139.62
LA31	1784	0	877.20	1784	0	∼	∼	∼	**1784**	1785.80	0	68.36
LA32	1850	0	837.20	1850	0	∼	∼	∼	**1850**	1850.23	0	56.95
LA33	1719	0	856.50	1719	0	∼	∼	∼	**1719**	1723.30	0	59.62
LA34	1721	0	833.20	1721	0	∼	∼	∼	**1721**	1721.40	0	62.75
LA35	1888	0	842.60	1888	0	∼	∼	∼	**1888**	1888.00	0	61.85
LA36	1311	3.39	387.50	1279	0.87	∼	∼	∼	**1268**	1300.47	0	231.98
LA37	∼	∼	∼	1411	1.00	∼	∼	∼	**1397**	1443.50	0	218.25
LA38	∼	∼	∼	1208	1.00	∼	∼	∼	1203	1235.70	0.59	213.25
LA39	∼	∼	∼	1236	0.24	∼	∼	∼	1235	1252.67	0.16	209.32
LA40	∼	∼	∼	1225	0.25	∼	∼	∼	**1222**	1247.37	0	231.58
ORB01	∼	∼	∼	∼	∼	1059	0	∼	**1059**	1069.35	0	69.25
ORB05	∼	∼	∼	∼	∼	887	0	∼	**887**	892.80	0	75.24
ORB10	∼	∼	∼	∼	∼	944	0	∼	**944**	956.25	0	71.29
ABZ5	∼	∼	∼	∼	∼	1234	0	∼	**1234**	1242.52	0	85.25
ABZ6	∼	∼	∼	∼	∼	943	0	∼	**943**	948.25	0	95.62
ABZ8	∼	∼	∼	∼	∼	669	0.60	∼	**669**	681.10	0.60	326.24
ABZ9	∼	∼	∼	∼	∼	683	0.59	∼	685	692.23	0.88	335.24
NIS	39			43		40			50			
BKS found	23			34		37			46			
Ratio (%)	58.97			79.07		92.50			92.00			

Instance	rHEA				bHEA			
	EBS	Avg.	RPE	CT	EBS	Avg.	RPE	CT
LA22	**927**	937.47	0	113.74	**927**	949.77	0	99.69
LA23	**1032**	1032.00	0	89.95	**1032**	1032.00	0	85.02
LA24	**935**	951.13	0	113.63	**935**	960.77	0	103.51
LA25	**977**	992.87	0	111.98	**977**	1003.30	0	105.13
LA26	**1218**	1222.53	0	85.36	**1218**	1243.07	0	91.36
LA27	**1235**	1268.30	0	109.38	**1235**	1299.80	0	150.23
LA28	**1216**	1230.83	0	125.36	**1216**	1268.90	0	165.32
LA29	**1152**	1213.87	0	150.98	**1152**	1256.23	0	162.12
LA30	**1355**	1355.23	0	135.62	**1355**	1358.33	0	112.61
LA31	**1784**	1789.60	0	72.12	**1784**	1785.97	0	65.23
LA32	**1850**	1851.80	0	68.36	**1850**	1852.87	0	42.85
LA33	**1719**	1722.33	0	55.29	**1719**	1724.20	0	55.98
LA34	**1721**	1722.73	0	71.95	**1721**	1747.23	0	85.36
LA35	**1888**	1894.20	0	68.36	**1888**	1894.50	0	92.39
LA36	1269	1304.70	0.08	228.23	1269	1317.60	0.08	212.32
LA37	**1397**	1441.00	0	221.74	1401	1461.20	0.29	185.14
LA38	**1202**	1235.20	0.50	195.20	1203	1251.47	0.59	196.56

Table 6 continued

Instance	rHEA				bHEA			
	EBS	Avg.	RPE	CT	EBS	Avg.	RPE	CT
LA39	**1234**	1254.43	0.08	216.17	1239	1278.13	0.49	225.15
LA40	1222	1237.20	0	265.32	1225	1257.93	0.25	218.29
ORB01	**1059**	1072.52	0	85.25	**1059**	1075.25	0	91.24
ORB05	**887**	896.25	0	75.25	**887**	898.35	0	74.52
ORB10	**944**	957.62	0	65.25	**944**	962.25	0	68.25
ABZ5	**1234**	1243.25	0	79.36	**1234**	1247.25	0	87.25
ABZ6	**943**	947.85	0	85.54	**943**	954.25	0	75.65
ABZ8	670	682.21	0.75	298.35	**669**	683.52	0.60	315.24
ABZ9	**683**	691.24	0.59	341.20	685	693.39	0.88	326.41
NIS	50				50			
BKS found	45				43			
Ratio (%)	90.00				86.00			

[bold] indicates that the proposed algorithms obtained the best possible results compared to the comparison algorithms.

Table 7 Improvement comparison of HEAs with 13 algorithms

Algorithm	NIS	OA(%)	sHEA		rHEA		bHEA	
			aRPE(%)	RPI (%)	aRPE(%)	RPI (%)	aRPE(%)	RPI (%)
F&F	43	0.28	0.02	93.86	0.02	94.57	0.04	86.18
MA	43	0.33	0.02	94.73	0.02	95.34	0.04	88.15
DESS	43	1.71	0.02	98.98	0.02	99.10	0.04	97.71
TLBO	20	0.83	0.01	98.49	0.00	99.52	0.02	98.01
MGA	46	0.27	0.02	94.05	0.01	94.74	0.04	86.61
CEBFO	23	0.10	0.02	81.72	0.02	83.82	0.04	58.85
aLSGA	40	0.82	0.00	100.00	0.00	100.00	0.00	100.00
NS-HDE/EDA	43	1.29	0.02	98.55	0.02	98.71	0.04	96.73
NIMGA	45	0.75	0.02	97.80	0.01	98.05	0.04	95.04
pABC	38	0.03	0.00	100.00	0.00	100.00	0.00	100.00
GWO	39	1.23	0.00	100.00	0.00	99.84	0.00	99.84
PBA	43	0.19	0.02	90.62	0.02	91.69	0.04	78.88
WOA-LFDE	40	0.05	0.04	24.68	0.03	31.99	0.04	24.68

2015), modified GA (MGA)(Thammano & Teekeng, 2015) and new island model GA (NIMGA) (Kurdi, 2016).

The reported optimization results of these 13 algorithms and the optimization results of the proposed algorithms for solving 50 benchmark instances of JSSP are presented in Table 6. This table reports the average makespan of 30 independent runs of the developed algorithms to show the repeatability and mark the best performance for each instance. At the bottom of Table 6, NIS, BKS found, Ratio (%) are reported for each algorithm. BKS to NIS ratio for sHEA, rHEA and bHEA are 92.00%, 90.00%, 86.00%, respectively, while the highest ratio of finding BKS to NIS is 94.74% for Asadzadeh (2016); however, the authors have only solved a smaller set of 38 problems. The second best BKS to NIS is 92.50% for Liu et al. (2020) who have solved 44 of 50 problems. However, sHEA, rHEA and bHEA can optimally solve 46, 45 and 43 out of 50 problems, respectively, and the instances that can not be solved to their BKS values are widely considered hard, and a few researchers were able to solve.

To explicitly demonstrate the proposed algorithms' performance, the respective RPI(%) for proposed algorithms is calculated by only considering respective EBS for reported instances for each algorithm. The RPI(%), as reported in Table 7, clearly shows that proposed algorithms (i.e., sHEA, rHEA and rHEA) outperform the 13 published approaches, specially pABC (Asadzadeh, 2016) and WOA-LFDE (Liu et al., 2020), which show best-performing algorithms in terms of ratio of finding BKS to NIS.

To reinforce the competitiveness of the HEAs (i.e., sHEA, rHEA and bHEA), additional 9 published approaches are considered. These algorithms are on the basis of heuristics

Table 8 Comparison of HEAs with another 9 approaches based on aRPE

Instance	Number of instances	Size	Average relative percentage error (aRPE)											
			SB	GRASP	Parallel GRASP	MA_1	MA_2	MA_3	CULT	UPLA	GPSO	sHEA	rHEA	bHEA
ft06	1	6 × 6	0	0	0	∼	∼	∼	0	0	0	**0**	**0**	**0**
ft10	1	10 × 10	0	0.86	0	∼	∼	∼	0	0	0	**0**	**0**	**0**
ft20	1	20 × 5	1.12	0.34	0	∼	∼	∼	0	0	0	**0**	**0**	**0**
la01-la05	5	10 × 5	0.8	0.23	0	0	0	0	0	0	0	**0**	**0**	**0**
la06-la10	5	15 × 5	0.02	0	0	0	0	0	0	0	0	**0**	**0**	**0**
la11-la15	5	20 × 5	0	0	0	0	0	0	0	0	0	**0**	**0**	**0**
la16-la20	5	10 × 10	1.73	0.13	0	0.11	0.11	0	0.11	0	0	**0**	**0**	**0**
la21-la25	5	15 × 10	1.79	3.54	0.76	2.14	0.7	0.41	0.78	0.35	0	**0**	**0**	**0**
la26-la30	5	20 × 10	3.08	6.15	1.58	3.18	1.34	1.06	1.01	1.02	0.83	**0**	**0**	**0**
la31-la35	5	30 × 10	0	0.37	0	0	0	0	0	0	0.04	**0**	**0**	**0**
la36-la40	5	15 × 15	3.36	4.62	1.46	3.23	1.3	0.68	1.32	1.01	1.5	0.15	**0.13**	0.34

[bold] indicates that the proposed algorithms obtained the best possible results compared to the comparison algorithms.

Table 9 Comparison of the 4 algorithms with the proposed algorithms for JSSP (Taillard's instances)

Instance	Size	NIS	Average relative percentage error (aRPE)						
			NS-HDE/EDA	PSO	GA	MCDE/TS	sHEA	rHEA	bHEA
Ta01-10	15 × 15	10	4.49	13.98	12.07	2.66	**0.85**	0.89	1.62
Ta11-20	20 × 15	10	8.82	16.24	16.07	3.65	**2.95**	3.01	3.72
Ta21-30	20 × 20	10	9.91	18.27	16.2	4.07	3.73	**3.62**	4.35
Ta31-40	30 × 15	10	6.54	18.1	15.71	5.25	**4.29**	4.43	5.98
Ta41-50	30 × 20	10	7.65	18.38	14.55	7.54	6.14	**6.01**	9.14
Ta51-60	50 × 15	10	2.91	16.95	13.97	0.82	**0.65**	0.71	1.35
Ta61-70	50 × 20	10	4.36	16.48	14.27	3.40	2.89	**2.76**	3.43
Ta71-80	100 × 20	10	3.01	15.61	14.38	0.68	**0.61**	1.01	1.19

[bold] indicates that the proposed algorithms obtained the best possible results compared to the comparison algorithms.

approach such as shifting bottleneck heuristic (SB) (Adams et al., 1988), cultural algorithm (CULT) (Cortés Rivera et al., 2007); adaptive search approach such as greedy randomized search procedure (GRASP) (Binato et al., 2002), parallel version of GRASP (parallel GRASP) (Aiex et al., 2003); strong memetic algorithms (MAs) such as MA_1 (Hasan et al., 2009), MA_2 (Raeesi & Kobti, 2011), MA_3 (Raeesi N & Kobti, 2012); hybrid approach such as PSO with GA operators (GPSO) (Abdel-Kader, 2018); two-level meta-heuristics approach such as upper-level algorithm and lower-level algorithm (UPLA) (Pongchairerks, 2019). The comparison of HEAs with these algorithms are performed for solving 43 problems, and aRPE of the equal-sized problem is reported in Table 8. Boldface indicates the better performance of the algorithms. The proposed approaches also show competitive results over the 9 reported algorithms.

Zhao et al. (2016) tested their proposed NS-HDE/EDA against standalone PSO and GA by solving the 80 Tallard's instances of JSSP. These results are adopted for testing our proposed algorithms against these baseline algorithms, including a multi-operator communication-based DE with sequential TS (MCDE/TS) algorithm (Mahmud et al., 2021). Table 9 shows that the proposed algorithms, specially sHEA and rHEA, outperform the listed algorithms.

Statistical analysis

The section identifies the statistical significance among the varied population structured-based HEAs (i.e., sHEA, rHEA and bHEA) before comparing between the best performing and existing algorithms. Since according to the Table 6 and Equation 23, the aRPE of the sHEA, rHEA and bHEA for solving 50 problems are 0.044%, 0.040% and 0.063%, respectively, which are pretty similar, no firm evidence can justify the supremacy of one population structure over another. Thus, two non-parametric tests (Friedman test and Wilcoxon sign-test) are first conducted to reinforce the decision, reflecting any statistical distinction in them. The Friedman test results are reported in Table 10, explicitly reflecting through the mean rank of the sHEA (1.84), rHEA

Table 10 Rank measurement of sHEA, rHEA and bHEA using the Friedman test

Algorithm	bHEA	rHEA	sHEA
Mean Rank	2.21	1.96	1.84

(1.96) and bHEA (2.21) that the sHEA dominates bHEA and rHEA (i.e., having lower mean rank compared to the other two). The Wilcoxon signed test is also implemented to analyse the differences between the best and near-best results according to Table 10. The results at 5% confidence level are reported in Table 11. The obtained Z values are: $Z_{sHEA-rHEA} = -0.946$ and $Z_{sHEA-bHEA} = -3.063$ and the p-values of these pairs are 0.344 and 0.002, respectively. Since the p-value of pair (sHEA-bHEA) is lower than 0.05, it illustrates the supremacy of sHEA over bHEA. However, no significant statistical difference can be found between sHEA and rHEA (since the p-value is higher than 0.05).

To further compare the robustness among these three algorithms to solve the varied-sized instances, a descriptive statistical analysis can be performed through a boxplot that is adopted to show the five-number summary (i.e., minimum, first quartile, second quartile, third quartile and maximum). The reported average makespan results of 30 replications in Table 6 are taken as samples for this test. The five-number summary of RPE ($=\frac{Avg.-BKS}{BKS}$) in Fig. 7 shows that the minimum and the first and second quartiles of sHEA coincide at 0 RPE, i.e., the algorithm can find the BKS in every replication for 50% of the varied-sized instances considered (standard deviations among the replications are zero). Although the interquartile range (IQR) of sHEA and rHEA are almost equal, bHEA's results are highly dispersed. The maximum value of sHEA is lower than both algorithms (rHEA and bHEA). Thus, sHEA is more robust in solving varied-sized instances than rHEA and bHEA.

Since the sHEA performs better than rHEA and bHEA, these two non-parametric tests are conducted again between sHEA and 11 other algorithms, excluding the CEBFO and TLBO algorithms. These algorithms are excluded because solving less than 30 instances is statistically insignificant for these tests. The Friedman test results are reported in Table 12, reflecting that sHEA is highly competitive and consistent among all statistically dispersed results. The Wilcoxon signed- test is performed to determine any substantial dominance of pABC and WOA-LFDE over sHEA. The p-values of sHEA-pABC and sHEA-WOA-LFDE are 0.157 and 0.891, respectively. Both values are significantly higher than 0.05, and thus, neither pABC nor WOA-LFDE is better than sHEA. Meanwhile, the Z-value of these pairs are: $Z_{sHEA-pABC} = -1.414$ and $Z_{sHEA-WOA-LFDE} = -0.137$. Therefore, sHEA is highly competitive and not outperformed by any other algorithms.

However, this JSSP resembles many real-life problems, such as multi-product assembly of the apparel industries (Guo et al., 2006), vehicle production in automobile industries (Zhang et al., 2013), assembly fixtures in aeronautic industries (Da Silva et al., 2014) and customized products in furniture industries (Vidal et al., 2011). Several schedulers exist and are used in industries that mainly sequence customer orders considering demand, material availability, and due date. However, the proposed algorithm (i.e., sHEA) can optimize workflows for each job/customer order (that comprises a set of operations) by correctly mapping interrelated resources, resulting in a holistic system overview of the production insight. A schedule includes the start and the finish time of each job/customer order, their sequences in machines, and machine idle times. This predictive visibility and insight delivered by the JSSP explanations allow decision-makers adequate time and maneuverability to improve operational decisions to leverage benefits, integrating these JSSP results to material acquisition (Sawik, 2016), due date assignment (Steiner & Zhang, 2011), order acceptance (Sarvestani et al., 2019), and delivery planning problems (Ullrich, 2013). Overall, it can create and foster a responsive relation between supply chain stages.

Conclusion and recommendation

This paper developed HEAs, in which different population structures in a multi-algorithmic framework with a performance-driven switching mechanism are proposed to solve the computationally intractable NP-hard JSSPs. Algorithms may perform poorly over generations, and continuing evolution in those algorithms is unrealistic. Thus, the capabilities of multiple algorithms were integrated into a single algorithmic scheme, and a performance-driven meta-heuristic switching approach was proposed to emphasize the best-performing algorithms. The initial population impacts the evolutionary process, and thus this study employed the MSS with equally prioritizing diversity and solution quality of schedules and the population diversity was maintained over the evolutions by implementing the diversity check mechanism. EAs explore a search space utilizing a population, and dividing the population into sub-populations may destroy the search uniformity and exploration capability. Thus, this study developed three approaches (i.e., sHEA, rHEA and bHEA) to illustrate the impacts of different population structures. These algorithms' exploitation property was enhanced by employing the TS inspired local search techniques. This local search utilized the N7 neighbourhood structure and recorded the moves for a specific tenure to avoid reevaluating the same solutions and becoming trapped in local optima.

An extensive experiment was conducted to test the performance of our three developed hybrid approaches against their

Table 11 Wilcoxon signed rank test between the best and near-best algorithms (sHEA, rHEA and bHEA)

		Ranks			Test statistics	
		N	Mean rank	Sum of ranks	Z	p-value
sHEA–rHEA	Negative ranks	3	10.67	32.00	− 0.946	0.344
	Positive ranks	10	5.90	59.00		
	Ties	45	∼	∼		
	Total	58	∼	∼		
bHEA–sHEA	Negative ranks	0	0	0	− 3.063	0.002
	Positive ranks	12	6.50	78.00		
	Ties	46	∼	∼		
	Total	58	∼	∼		

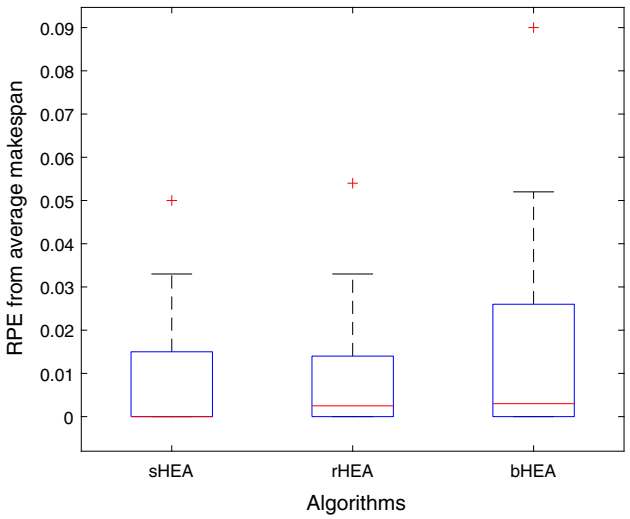


Fig. 7 Robustness of the developed algorithms

component algorithms (i.e., TS, DE and PSO), as well as relevant 26 popular algorithms. The bHEA could not produce promising results compared to rHEA and sHEA. However, the computational results and the statistical analysis have shown that rHEA and sHEA are highly competitive compared to any algorithms reviewed in this paper. Although there is no statistically significant difference between rHEA and sHEA, the computational results showed that sHEA is 1.1% better than rHEA based on Equation 24. Thus, sHEA is the most competitive proposed approach in this paper. The strategy implemented in sHEA showed better exploration property at the start of the evolutionary process and better exploitation property at the later stage, increasing the chance of finding global optima. Moreover, the statistical analysis showed that the proposed sHEA ranked first with the mean value of 5.09 compared to the near-best algorithms of the existing 26 algorithms.

Although the proposed sHEA produces promising results, being highly competitive with other algorithms, there is scope for improvement. Thus, the main limitations of the current work and some suggestions for future works include the following:

- MODE variants were selected based on their characteristics, which may not be the best combinations. Thus, mutation operators can be investigated more definitively.
- The solution quality controlled the switching from an under-performing algorithm to other possibilities, excluding the population diversity, although the population diversity is ensured using the diversity check mechanism. Both population diversity and solution quality can be considered together while switching decision is made.
- Although the proposed algorithms' effectiveness was assessed against a list of existing algorithms, the switching strategy's contribution can be further assessed compared with the most popular reinforcement learning.
- The Tabu list and diversity check mechanism was introduced to prevent revisiting the identical solutions. Although Tabu list overcame the problem to some extent; however, it can not record all the moves. Thus, the memory-based approach, where every move can be recorded for a few generations, can be an exciting approach to be explored. It would prevent revisiting the same solution and becoming trapped in local optima, reducing computational expenses.
- The classical JSSP can be extended to supply chain scheduling, including the vehicle routing problems and supply portfolio decisions and then the effectiveness of the proposed algorithm can be assessed.

The authors are currently pursuing a number of these directions in other works.

Table 12 Rank measurement between proposed and state-of-the-art algorithms using Friedman test

Algorithm	F&F	MA	DESS	MGA	aLSGA	NS-HDE/EDA	NIMGA	pABC	GWO	PBA	WOA-LFDE	sHEA
Mean rank	6.18	5.73	8.20	5.64	7.59	8.23	6.82	5.29	8.53	5.56	5.15	5.09

Author Contributions All authors have written this paper and have done the research which support it.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Abdel-Kader, R. F. (2018). An improved PSO algorithm with genetic and neighborhood-based diversity operators for the job shop scheduling problem. *Applied Artificial Intelligence, 32*, 433–462.

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science, 34*, 391–401.

Ahmadian, M. M., Salehipour, A., & Cheng, T. (2021). A meta-heuristic to solve the just-in-time job-shop scheduling problem. *European Journal of Operational Research, 288*, 14–29.

Aiex, R. M., Binato, S., & Resende, M. G. (2003). Parallel GRASP with path-relinking for job shop scheduling. *Parallel Computing, 29*, 393–430.

Akram, K., Kamal, K., & Zeb, A. (2016). Fast simulated annealing hybridized with quenching for solving job shop scheduling problem. *Applied Soft Computing, 49*, 510–523.

Applegate, D., & Cook, W. (1991). A computational study of the job-shop scheduling problem. *ORSA Journal on computing, 3*, 149–156.

Asadzadeh, L. (2015). A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Computers & Industrial Engineering, 85*, 376–383.

Asadzadeh, L. (2016). A parallel artificial bee colony algorithm for the job shop scheduling problem with a dynamic migration strategy. *Computers & Industrial Engineering, 102*, 359–367.

Asadzadeh, L., & Zamanifar, K. (2010). An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. *Mathematical and Computer Modelling, 52*, 1957–1965.

Baykasoğlu, A., Hamzadayi, A., & Köse, S. Y. (2014). Testing the performance of teaching-learning based optimization (TLBO) algorithm on combinatorial problems: Flow shop and job shop scheduling cases. *Information Sciences, 276*, 204–218.

Binato, S., Hery, W., Loewenstern, D., & Resende, M. G. (2002). A GRASP for job shop scheduling. In *Essays and surveys in metaheuristics* (pp. 59–79). Springer.

Brucker, P., Jurisch, B., & Sievers, B. (1994). A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics, 49*, 107–127.

Çaliş, B., & Bulkan, S. (2015). A research survey: Review of AI solution strategies of job shop scheduling problem. *Journal of Intelligent Manufacturing, 26*, 961–973.

Chen, T., Tang, K., Chen, G., & Yao, X. (2012). A large population size can be unhelpful in evolutionary algorithms. *Theoretical Computer Science, 436*, 54–70.

Cheng, T., Peng, B., & Lü, Z. (2016). A hybrid evolutionary algorithm to solve the job shop scheduling problem. *Annals of Operations Research, 242*, 223–237.

Cortés Rivera, D., Landa Becerra, R., & Coello Coello, C. A. (2007). Cultural algorithms, an alternative heuristic to solve the job shop scheduling problem. *Engineering Optimization, 39*, 69–85.

Cruz-Chávez, M. A., Rosales, M. H. C., Zavala-Diaz, J. C., Aguilar, J. A. H., Rodríguez-León, A., Avelino, J. C. P., Ortiz, M. E. L., & Salinas, O. H. (2019). Hybrid micro genetic multi-population algorithm with collective communication for the job shop scheduling problem. *IEEE Access, 7*, 82358–82376.

Da Silva, B. J. V., Morabito, R., Yamashita, D. S., & Yanasse, H. H. (2014). Production scheduling of assembly fixtures in the aeronautical industry. *Computers & Industrial Engineering, 67*, 195–203.

Dao, T.-K., Pan, T.-S., Nguyen, T.-T., & Pan, J.-S. (2018). Parallel bat algorithm for optimizing makespan in job shop scheduling problems. *Journal of Intelligent Manufacturing, 29*, 451–462.

Elsayed, S. M., Sarker, R. A., & Essam, D. L. (2011). Multi-operator based evolutionary algorithms for solving constrained optimization problems. *Computers & Operations Research, 38*, 1877–1896.

Elsayed, S. M., Sarker, R. A., & Essam, D. L. (2012). On an evolutionary approach for constrained optimization problem solving. *Applied Soft Computing, 12*, 3208–3227.

Fisher, H. (1963). Probabilistic learning combinations of local job-shop scheduling rules. *Industrial Scheduling*, (pp. 225–251).

Gao, L., Zhang, G., Zhang, L., & Li, X. (2011). An efficient memetic algorithm for solving the job shop scheduling problem. *Computers & Industrial Engineering, 60*, 699–705.

Gao, S., Yu, Y., Wang, Y., Wang, J., Cheng, J., & Zhou, M. (2019). Chaotic local search-based differential evolution algorithms for optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 51*, 3954–3967.

Ge, H.-W., Sun, L., Liang, Y.-C., & Qian, F. (2008). An effective PSO and AIS-based hybrid intelligent algorithm for job-shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 38*, 358–368.

Giffler, B., & Thompson, G. L. (1960). Algorithms for solving production-scheduling problems. *Operations Research, 8*, 487–503.

Goldberg, D. E. (2006). *Genetic algorithms*. Pearson Education India.

Goli, A., Tirkolaee, E. B., & Aydın, N. S. (2021). Fuzzy integrated cell formation and production scheduling considering automated

guided vehicles and human factors. *IEEE Transactions on Fuzzy Systems, 29*, 3686–3695.

Guo, Z., Wong, W. K., Leung, S.Y.-S., Fan, J., & Chan, S. (2006). Mathematical model and genetic optimization for the job shop scheduling problem in a mixed-and multi-product assembly environment: A case study based on the apparel industry. *Computers & Industrial Engineering, 50*, 202–219.

Hasan, S. K., Sarker, R., Essam, D., & Cornforth, D. (2009). Memetic algorithms for solving job-shop scheduling problems. *Memetic Computing, 1*, 69–83.

Ibrahim, A. M., & Tawhid, M. A. (2022). An improved artificial algae algorithm integrated with differential evolution for job-shop scheduling problem. *Journal of Intelligent Manufacturing*, (pp. 1–16).

Islam, M. A., Gajpal, Y., & ElMekkawy, T. Y. (2021). Hybrid particle swarm optimization algorithm for solving the clustered vehicle routing problem. *Applied Soft Computing, 110*, 107655.

Jiang, T., & Zhang, C. (2018). Application of grey wolf optimization for solving combinatorial problems: Job shop and flexible job shop scheduling cases. *IEEE Access, 6*, 26231–26240.

Kannan, V., & Ghosh, S. (1993). An evaluation of the interaction between dispatching rules and truncation procedures in job-shop scheduling. *The International Journal of Production Research, 31*, 1637–1654.

Kurdi, M. (2016). An effective new island model genetic algorithm for job shop scheduling problem. *Computers & Operations Research, 67*, 132–142.

Lawrence, S. (1984). *Resouce constrained project scheduling: An experimental investigation of heuristic scheduling techniques (supplement)*. Graduate School of Industrial Administration: Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Lin, T.-L., Horng, S.-J., Kao, T.-W., Chen, Y.-H., Run, R.-S., Chen, R.-J., Lai, J.-L., & Kuo, I.-H. (2010). An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications, 37*, 2629–2636.

Liu, F., Qi, Y., Xia, Z., & Hao, H. (2009). Discrete differential evolution algorithm for the job shop scheduling problem. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation* (pp. 879–882). ACM.

Liu, M., Yao, X., & Li, Y. (2020). Hybrid whale optimization algorithm enhanced with Lévy flight and differential evolution for job shop scheduling problems. *Applied Soft Computing, 87*, 105954.

Mahmud, S., Abbasi, A., Chakrabortty, R. K., & Ryan, M. J. (2021). Multi-operator communication based differential evolution with sequential tabu search approach for job shop scheduling problems. *Applied Soft Computing, 108*, 107470.

Marichelvam, M., Geetha, M., & Tosun, Ö. (2020). An improved particle swarm optimization algorithm to solve hybrid flowshop scheduling problems with the effect of human factors-a case study. *Computers & Operations Research, 114*, 104812.

Meeran, S., & Morshed, M. (2014). Evaluation of a hybrid genetic tabu search framework on job shop scheduling benchmark problems. *International Journal of Production Research, 52*, 5780–5798.

Mishra, S., Bose, P., & Rao, C. (2017). An invasive weed optimization approach for job shop scheduling problems. *The International Journal of Advanced Manufacturing Technology, 91*, 4233–4241.

Onwubolu, G., & Davendra, D. (2006). Scheduling flow shops using differential evolution algorithm. *European Journal of Operational Research, 171*, 674–692.

Peng, B., Lü, Z., & Cheng, T. (2015). A tabu search/path relinking algorithm to solve the job shop scheduling problem. *Computers & Operations Research, 53*, 154–164.

Pezzella, F., & Merelli, E. (2000). A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research, 120*, 297–310.

Pongchairerks, P. (2019). A two-level metaheuristic algorithm for the job-shop scheduling problem. *Complexity*, *2019*.

Ponnambalam, S., Aravindan, P., & Rao, P. S. (2001). Comparative evaluation of genetic algorithms for job-shop scheduling. *Production Planning & Control, 12*, 560–574.

Ponsich, A., & Coello, C. A. C. (2013). A hybrid differential evolution-tabu search algorithm for the solution of job-shop scheduling problems. *Applied Soft Computing, 13*, 462–474.

Ponsich, A., Tapia, M. G. C., & Coello, C. A. C. (2009). Solving permutation problems with differential evolution: An application to the jobshop scheduling problem. In *2009 ninth international conference on intelligent systems design and applications* (pp. 25–30). IEEE.

Pratchayaborirak, T., & Kachitvichyanukul, V. (2011). A two-stage PSO algorithm for job shop scheduling problem. *International Journal of Management Science and Engineering Management, 6*, 83–92.

Qian, W., & Li, M. (2018). Convergence analysis of standard particle swarm optimization algorithm and its improvement. *Soft Computing, 22*, 4047–4070.

Qiu, X., & Lau, H. Y. (2014). An AIS-based hybrid algorithm for static job shop scheduling problem. *Journal of Intelligent Manufacturing, 25*, 489–503.

Raeesi, N. M. R., & Kobti, Z. (2011). A machine operation lists based memetic algorithm for job shop scheduling. In *2011 IEEE congress of evolutionary computation (CEC)* (pp. 2436–2443). IEEE.

Raeesi, N., M, R., & Kobti, Z. (2012). A memetic algorithm for job shop scheduling using a critical-path-based local search heuristic. *Memetic Computing, 4*, 231–245.

Rahman, H. F., Chakrabortty, R. K., & Ryan, M. J. (2020). Memetic algorithm for solving resource constrained project scheduling problems. *Automation in Construction, 111*, 103052.

Rego, C., & Duarte, R. (2009). A filter-and-fan approach to the job shop scheduling problem. *European Journal of Operational Research, 194*, 650–662.

Ren, Q.-d-e-j, & Wang, Y. (2012). A new hybrid genetic algorithm for job shop scheduling problem. *Computers & Operations Research, 39*, 2291–2299.

Sallam, K. M., Chakrabortty, R. K., & Ryan, M. J. (2020). A two-stage multi-operator differential evolution algorithm for solving resource constrained project scheduling problems. *Future Generation Computer Systems, 108*, 432–444.

Sallam, K. M., Elsayed, S. M., Sarker, R. A., & Essam, D. L. (2017). Landscape-based adaptive operator selection mechanism for differential evolution. *Information Sciences, 418*, 383–404.

Sarvestani, H. K., Zadeh, A., Seyfi, M., & Rasti-Barzoki, M. (2019). Integrated order acceptance and supply chain scheduling problem with supplier selection and due date assignment. *Applied Soft Computing, 75*, 72–83.

Sawik, T. (2016). Integrated supply, production and distribution scheduling under disruption risks. *Omega, 62*, 131–144.

Sha, D., & Hsu, C.-Y. (2006). A hybrid particle swarm optimization for job shop scheduling problem. *Computers & Industrial Engineering, 51*, 791–808.

Sharma, N., Sharma, H., & Sharma, A. (2018). Beer froth artificial bee colony algorithm for job-shop scheduling problem. *Applied Soft Computing, 68*, 507–524.

Steiner, G., & Zhang, R. (2011). Minimizing the weighted number of tardy jobs with due date assignment and capacity-constrained deliveries. *Annals of Operations Research, 191*, 171–181.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64*, 278–285.

Tasgetiren, M. F., & Suganthan, P. N. (2006). A multi-populated differential evolution algorithm for solving constrained optimization problem. In *2006 ieee international conference on evolutionary computation* (pp. 33–40). IEEE.

Thammano, A., & Teekeng, W. (2015). A modified genetic algorithm with fuzzy roulette wheel selection for job-shop scheduling problems. *International Journal of General Systems, 44*, 499–518.

Ullrich, C. A. (2013). Integrated machine scheduling and vehicle routing with time windows. *European Journal of Operational Research, 227*, 152–165.

Vidal, J. C., Mucientes, M., Bugarín, A., & Lama, M. (2011). Machine scheduling in custom furniture industry through neuro-evolutionary hybridization. *Applied Soft Computing, 11*, 1600–1613.

Wang, F., Zhang, H., Li, K., Lin, Z., Yang, J., & Shen, X.-L. (2018). A hybrid particle swarm optimization algorithm using adaptive learning strategy. *Information Sciences, 436*, 162–177.

Wang, X., & Duan, H. (2014). A hybrid biogeography-based optimization algorithm for job shop scheduling problem. *Computers & Industrial Engineering, 73*, 96–114.

Wang, Z., Zhang, J., & Yang, S. (2019). An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals. *Swarm and Evolutionary Computation, 51*, 100594.

Wisittipanich, W., & Kachitvichyanukul, V. (2012). Two enhanced differential evolution algorithms for job shop scheduling problems. *International Journal of Production Research, 50*, 2757–2773.

Zarandi, M. H. F., Asl, A. A. S., Sotudian, S., & Castillo, O. (2020). A state of the art review of intelligent scheduling. *Artificial Intelligence Review, 53*, 501–593.

Zhang, C., Li, P., Guan, Z., & Rao, Y. (2007). A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research, 34*, 3229–3242.

Zhang, J., Ding, G., Zou, Y., Qin, S., & Fu, J. (2019). Review of job shop scheduling research and its new perspectives under industry 4.0. *Journal of Intelligent Manufacturing, 30*, 1809–1830.

Zhang, R., Song, S., & Wu, C. (2013). A hybrid artificial bee colony algorithm for the job shop scheduling problem. *International Journal of Production Economics, 141*, 167–178.

Zhao, F., Jiang, X., Zhang, C., & Wang, J. (2015). A chemotaxis-enhanced bacterial foraging algorithm and its application in job shop scheduling problem. *International Journal of Computer Integrated Manufacturing, 28*, 1106–1121.

Zhao, F., Qin, S., Yang, G., Ma, W., Zhang, C., & Song, H. (2018). A differential-based harmony search algorithm with variable neighborhood search for job shop scheduling problem and its runtime analysis. *IEEE Access, 6*, 76313–76330.

Zhao, F., Shao, Z., Wang, J., & Zhang, C. (2016). A hybrid differential evolution and estimation of distribution algorithm based on neighbourhood search for job shop scheduling problems. *International Journal of Production Research, 54*, 1039–1060.