White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of Leeds.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/247101/

Version: Accepted Version

Article:
Cui, M., Przewozniczek, M., Chen, H. et al. (2026) An island-based optimization framework for process planning and scheduling using migrations with empirical linkage. The Journal of Systems Architecture: Embedded Software Design, 179. 103931. ISSN: 1383-7621

https://doi.org/10.1016/j.sysarc.2026.103931

This is an author produced version of an article published in The Journal of Systems Architecture: Embedded Software Design, made available via the University of Leeds Research Outputs Policy under the terms of the Creative Commons Attribution License (CC-BY), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

Reuse
This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here: https://creativecommons.org/licenses/

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

UNIVERSITY OF LEEDS
University of Sheffield
UNIVERSITY of York

An Island-based Optimization Framework for Process Planning and Scheduling using Migrations with Empirical Linkage

Mingyue Cui†, Michal Przewozniczek‡, Haoran Chen†, Piotr Dziurzanski§, Marcin Komarnicki‡, Leandro Soares Indrusiak‖, Zhi Zhou†, Deke Guo†, Xu Chen†, Shuai Zhao†*

†Sun Yat-sen University, China ‡Wroclaw University of Science and Technology, Poland
§West Pomeranian University of Technology in Szczecin, Poland ‖University of Leeds, UK

***Abstract*—The integrated process planning and scheduling problem is widely present in the industrial field, which is recognised as an NP-hard problem and is often tackled by optimisation approaches such as Genetic Algorithms (GA). To provide sufficient computational resources that are often not available in real-world scenarios, a cloud-based GA framework with the island model (IM) is applied, in which the population is divided into several subpopulations (i.e., islands) that are evolved independently. However, existing IMs either do not support inter-island communication or require synchronisation of all islands with a non-trivial cost, which significantly undermines the optimisation quality. This paper presents a novel asynchronous island-based optimisation framework for integrated process planning and scheduling. The asynchronous IM eliminates the synchronisation cost due to inter-island communication via migrations and supports a wide range of migration strategies to improve optimisation efficiency. Two types of migration operators are constructed. The first, classical one, migrates a complete individual (i.e., individual-based migration). In the second approach, a finer-grained migration is constructed that migrates only the highly-dependent groups of genes by discovering the dependencies between genes in an individual using empirical linkage learning. This approach leads to slower convergence but is less likely to get trapped in local optima, which is significantly more effective than IMs using any of the considered individual-based migration operators. Evaluation based on a real-world manufacturing process planning and scheduling problem shows that the proposed method significantly outperforms existing methods.**

I. Introduction

The integrated process planning and scheduling problem constitutes a fundamental challenge in the Industry 4.0 paradigm, particularly when addressing decision-making problems that involve optimizing conflicting objectives such as performance, cost, and resource utilization [1], [2]. As an NP-hard problem, it is commonly addressed using metaheuristics, particularly Genetic Algorithms (GAs) [3]. In industrial practice, however, the computational demand for high-quality solutions often exceeds the capacity of local resources, especially when planning must be performed dynamically in response to orders or disruptions [4], [5]. This gap between the need for powerful optimization and the limitations of on-premise computing motivates the adoption of distributed computing paradigms, which is well-suited for larger-scale EFJSP problems and scenarios where optimization is invoked repeatedly as a reusable cloud service.

Cloud computing offers a viable platform to deliver the necessary computational power in an elastic and cost-effective manner [6]. This has enabled the adoption of distributed optimization models, among which the island model (IM) is a prominent paradigm for parallelising GAs [4], [7]–[9]. In the IM, the population is partitioned into subpopulations evolved independently on separate islands, with occasional migration between them to foster diversity and convergence. Despite its potential, conventional IMs suffer from a major limitation that they either do not allow inter-island communication [10], [11] or require synchronisation of all islands at fixed time points [4], [5], [12], [13]. For IMs without communications, each island maintains a relatively small population size and is likely to get stuck at local optima [4], [7]. The synchronisation enables communication between islands, but it imposes a significant cost, where early-finishing islands must wait for the late ones before they can communicate and continue. This undermines the scalability and efficiency of the optimization, especially in heterogeneous or dynamic cloud environments.

To overcome the synchronisation bottleneck, in our previous study [7], we propose: 1) a cloud-based GA optimization framework with asynchronous IM. Compared to the traditional IM [4], asynchronous IM removes the synchronisation burden and allows the exchange (i.e., migration) of individuals between islands at any time point; 2) a series of migration operators which migrate a complete individual based on predefined selection as well as replacement methods, and provide a comprehensive investigation of the effectiveness of the individual-based migrations; 3) a cluster-based architecture for deploying the proposed optimization framework on the cloud with a prototype implementation. The evaluation results demonstrated that inter-island communication is crucial for the effectiveness of the island-based GA framework. However, the individual-based migration provides a coarse-grained inter-island communication approach, which simply migrates a complete individual without the knowledge of inter-dependency in the genotype. As shown in [14], the inter-dependency between genes (i.e., the gene linkage) can be utilised to enhance the effectiveness of evolutionary algorithms, and is recognised as a key factor of the optimization performance [14]. However, existing linkage construction methods either cannot guarantee the correctness of the constructed gene linkage (i.e., the Statistical Linkage Learning) [15] or can only be applied to single-objective optimizations (i.e., the Empirical Linkage Learning) [16], [17]. This imposes a major obstacle towards effective inter-island communication for distributed island-based GA frameworks.

Contribution. This paper extends our previous conference paper [18] in the following aspects: 1) Unlike the predefined, blind selection/replacement strategies used in the conference version, we propose an empirical linkage learning method tailored to multi-objective manufacturing optimization problems, which can discover the gene interactions; 2) Instead of migrating complete individuals (as in the eight operators), we propose a linkage-based migration strategy that migrates only strongly dependent genes across the entire genotype; 3) We expand the experimental evaluation beyond the initial prototype implementation, providing a comprehensive evaluation of the proposed framework and migration strategies using two standard multi-objective quality indicators. The results demonstrate that our empirical linkage-based migration significantly outperforms existing methods in optimization performance. To sum up, the contributions of this paper are as follows:

- First, we develop an empirical linkage learning method dedicated to the considered multi-objective optimization problems, which can discover interdependencies among genes in a genotype of the multi-objective optimization problem.
- Second, a novel linkage-based migration strategy is proposed

*Corresponding author: Shuai Zhao, Email: zhaosh56@mail.sysu.edu.cn.
This work was supported by the National Natural Science Foundation of China (No. U25B2002 and No. 62302533), in part by the Guangdong Hong Kong Macao Applied Mathematics Center Project, and in part by the Guangxi Key R & D Program under Grant No.GuikeAB24010324.

that only migrates the highly dependent genes in the genotype. The exchanged set of gene values can be considered as building blocks [19]. The underlying intuition of linkage-based migration is that migrating only highly dependent gene subsets enables finer-grained communication and optimization at the building-block level rather than at the individual level, thereby improving the effectiveness of the search.
- Third, we substantially extend the experimental evaluation beyond the initial prototype, employ multi-objective quality indicators and compare the proposed linkage-based migration with the individual-based migrations and existing methods.

Based on the above contributions, the proposed linkage-based migration enables fine-grained inter-island communication by exchanging highly dependent gene groups rather than complete individuals. To the best of our knowledge, this is the first work to realize empirical linkage learning for multi-objective optimization and to support fine-grained communication in distributed GA frameworks.

The experiments are conducted based on a typical discrete manufacturing scenario (i.e., the production of distinct items) provided by our industrial partners, which is a major manufacturing type that can be found in most factories [4], [5], [8], [20]. The experimental results show that both individual-based and linkage-based migrations are effective compared to the traditional island-based GA. For the considered manufacturing optimization problem, the individual-based migration can provide competitive results with a relatively low computation cost, while the linkage-based migration provides the best optimization results among all evaluated methods, although with a higher computation time. With both migration approaches, we provide a reasonable balance for the trade-off between the optimization quality and computation cost, thereby catering to different application scenarios. The industrial partners that formulated these scenarios have implemented the pilot solutions of the proposed method, and have confirmed its applicability, generality and effectiveness [5], [7].

Organisation: This paper is organised as follows. Section II presents the related work. Section III describes the Multi-objective optimization problem. Section IV presents the asynchronous island-based GA framework and the individual-based migration approaches between islands. Section V proposes the linkage-based migration, with an example to illustrate its working mechanism. Finally, Section VI shows the experimental results while Section VII draws conclusions and highlights future work.

II. Related Work

Integrated process planning and scheduling constitutes a multi-objective optimization problem characterised by conflicting objectives. Consequently, the goal of the optimization process is to find a set of non-dominated solutions, referred to as the Pareto-optimal front, rather than a single solution. Methods for solving such problems primarily include evolutionary algorithms [7], [21] and approximate algorithms [22], [23]. This paper focuses on evolutionary algorithms due to their effectiveness in handling large-scale problems with discontinuous and or irregular solution spaces [8], [24], [25].

A. Evolutionary algorithms for Multi-objective Optimization

A Pareto-optimal solution can be determined using evolutionary algorithms, such as popular NSGA-II [26] or MOEA/D [21]. These algorithms follow a typical evolutionary algorithm scheme, in which populations are evolved in generations by applying crossover, mutation and selection operators. NSGA-II applies the concept of solution dominance explicitly as the solutions with lower dominance ranks are selected to form the population in the following generation. In case of comparing the solutions with the same dominance rank, the one with a higher *Crowding Distance* is selected, whereas the Crowding Distance is a diversity preserving mechanism [27].

In the MOEA/D algorithm proposed in [21], an M-objective function $F(x) = (f_1(x), \ldots, f_M(x))$ is decomposed into a set of single-objective subproblems, where the number of subproblems is an algorithm's parameter. The decomposition techniques can be based on the weighted sum, penalty-based boundary intersection, or Tchebycheff approaches. The subproblems can be solved by employing a single-objective evolutionary algorithm, for example with the one-point crossover operator and the standard mutation operator, as in the original MOEA/D proposal. An optimal solution to any of these subproblems is a part of Pareto-optimal vector of the original multi-objective problem $F(x)$. However, the subproblems are not solved independently, as MOEA/D benefits from the observation that the subproblems close to each other according to the Euclidean distance between their weight vectors tend to be similar. Thus, it is more beneficial to cross the individuals of neighboring subproblems, where the neighborhood size is treated as an algorithm's parameter. A population includes the best solutions to all subproblems $f_i(x)$ found during the optimization problem. With weight vectors fixed beforehand, diversity is inherently maintained and the costly Crowding-Distance calculation becomes unnecessary, e.g., the NSGA-II [26].

B. Island-based Optimization Framework on Cloud

The research into multi-population optimization frameworks is broad [10], [11], [28], [29]. To design effective Evolutionary Algorithms (EAs), it is necessary to preserve population diversity [19]. The Island Model (IM) has been proposed for reaching this objective [10], dividing the population into isolated groups called *islands*, each evolving independently. For instance, [11] presents a multi-population ant colony system for a bi-objective optimization problem, where each colony (island) minimises one objective. Communication between islands typically employs the *migration* operator, transferring the best individual from one island to another to enhance diversity. In [28], [29], advanced migration strategies are explored. [28] introduces the Gene Pattern Based Island Model (GePIM), which replaces traditional migration with Linked Gene Groups Migration (LGGM). LGGM exchanges building blocks—identified via a lightweight linkage learning method—instead of full individuals. Alternative migration methods are also proposed, such as probabilistic model exchange in [29].

Evolving multiple populations in a nearly independent manner enables parallel processing of each island [12]. In heterogeneous cloud environments, variable node latencies slow each GA generation; asynchronous GAs avoid global barriers and thus scale better [4], [13]. [30] evaluated several multi-objective evolutionary strategies for P2P networks, including diverse migration policies and island connectivity patterns. This paper investigates a similar approach, integrating advanced GA techniques with linkage learning for a real-world manufacturing problem. For distributed deployment, we use containerization as suggested in [31], orchestrated by Kubernetes. Each container hosts one GA island and communicates via the Redis database instead of the RabbitMQ message broker used in [31].

As described in Section I, the process planning and scheduling phases in smart factories have distinct computational demands compared to the manufacturing phase, making them ideal candidates for cloud-based optimization. To realize this potential, prior research [5], [9], [32], [33] has focused on developing the theoretical foundations and infrastructure for cloud-based island models in manufacturing. Leveraging a domain-specific ontology [32], a scalable cloud-ready digital twin was validated in [9]. Building upon this foundation, this paper focuses on designing a novel asynchronous island-based GA framework to enhance optimization performance.

C. Linkage Learning in Evolutionary Algorithms

An optimization problem is often expressed as $f(x) = \sum_j f_j(x_{I_j})$, where $f(\cdot)$ is the objective function, x is the genotype with a size of n (i.e., an individual solution), $f_j(\cdot)$ is a sub-objective that depends on certain genes in the individual, I_j is a subset of $\{1, ..., n\}$. The I_j subsets may be disjoint or not [34]. If two variables are the arguments of the same subfunction f_j, then we state that they are *directly* dependent. Genes without direct interaction can still be *indirectly* coupled through shared intermediaries.

In Gray-box optimization [14], the complete inter-dependencies between genes (variables) are known a priori, this knowledge is used to propose new recombination operators [35], [36] and perturbation operators for Iterated Local Search [37].

In Black-box optimization, the knowledge about gene dependencies remains unknown. However, it is as useful as in the Gray-box case in terms of the optimization quality. Linkage learning is a group of techniques for discovering gene dependencies in black-box optimization. Linkage learning is a part of many state-of-the-art GA-based optimizers for a wide range of problem domains, including binary problems [16], [17], non-binary discrete problems [38], [39], permutation-based problems [40], [41], and problems encoded by real-valued genotypes [42]. The evolutionary algorithms that employ linkage learning are also the state-of-the-art methods in solving multi-objective problems [24], [43].

To discover the gene dependencies, many state-of-the-art EAs use Statistical Linkage Learning (SLL) that is based on the idea of Dependency Structure Matrix (DSM) [44] (see Equation (1) where X and Y are random variables).

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} \tag{1}$$

From the pairwise-dependency DSM, hierarchical clustering produces a linkage tree whose internal nodes encode nested gene interactions. The main disadvantage of SLL is that it does not provide any guarantee for the correctness of the reported gene dependencies. That is, the SLL can report the *false linkage* in which the two genes are actually independent [16]. This flaw reduces optimizer efficacy [15]. Furthermore, DSM-using EAs were shown highly effective in solving multi-objective problems in binary domains [24], [43]. Note that multi-objective problems may be interpreted as a group of single-objective problems [43]. Thus, in multi-objective optimization, it is necessary to maintain many linkages that refer to various parts of the Pareto front. To obtain this, Multi-Objective GOMEA (MO-GOMEA) [43] clusters its population at every method's iteration concerning the part of the Pareto front they refer to. In Multi-Objective P3 (MO-P3) [24], the same objective (obtaining an effective individual mixing operator facilitating the optimization of various parts of the Pareto front) is obtained differently. Instead of clustering the population, MO-P3 takes advantage of the high linkage diversity inherited from P3, which is its core idea. Note that MO-GOMEA and MO-P3 have significantly outperformed the well-known multi-objective optimization EAs (e.g., NSGA-II [27] and MOEA/D [21]) on a wide set of theoretical problems [24], [43]. Their significant superiority was also observed for practical problems [24].

In addition, researchers proposed the Linkage Learning based on Local Optimization (3LO) method [16], as is shown in Figure 1. By perturbing a gene within an individual and employing synchronous hill-climbing optimization on both the original and perturbed individuals, this method identifies dependencies by comparing their optimized genotypes, thereby constructing high-quality linkage information. However, a major drawback of this method is the high computational cost of linkage discovery, which limits its applicability to large-scale real-world problems.

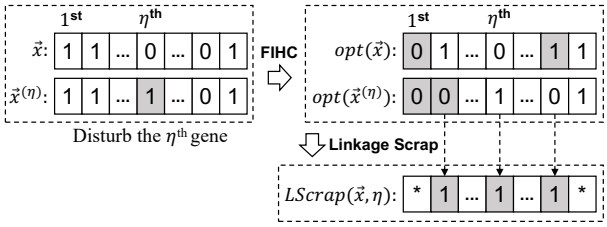


Fig. 1: An illustration of the 3L0 (the blocks in grey indicate genes that are changed in each step).

D. Exact, Matheuristic, and Learning-based Methods

Besides evolutionary algorithms, extended flexible job-shop scheduling problems have also been studied using non-GA approaches, including exact methods, matheuristic methods, and learning-based methods. These methods represent important research directions and can be effective under suitable modeling and computational assumptions. However, their applicability as direct baselines for the problem studied in this paper is not straightforward, because the considered problem combines heterogeneous discrete decisions, mutually exclusive shared manufacturing resources, and multi-objective evaluation through an analytical plant model.

Exact methods include mixed-integer linear programming (MILP) [45], constraint programming (CP) [46], and branch-and-bound/branch-and-cut based search [47]. These methods are valuable because they can provide optimality certificates, lower bounds, or optimality gaps for small and medium-sized instances when the scheduling problem can be expressed by a compact mathematical or logical formulation. They are therefore often used for model validation, benchmark generation, or solving simplified variants of job-shop and flexible job-shop scheduling problems. However, applying such methods to the problem considered in this paper would require a full solver-accessible reformulation of the analytical plant model. In particular, machine assignment, manufacturing-way selection, priority-induced sequencing, and mutually exclusive shared-resource constraints would have to be encoded using additional binary variables, disjunctive constraints, indicator constraints, big-M constraints, or CP global constraints. For a multi-objective setting, repeated scalarizations or other Pareto-front construction mechanisms would also be required. As the number of jobs, machines, manufacturing ways, and shared-resource conflicts increases, the resulting branch-and-bound or branch-and-cut search space can grow rapidly, making the wall-clock time required for repeatedly solving many instances impractical. Therefore, although exact solvers such as CPLEX [48] and OR-Tools [49] are useful for small-scale verification or bound computation, they are not adopted as direct computational baselines for the large-scale multi-objective experiments in this work.

Matheuristic methods combine mathematical programming with heuristic search to improve scalability [50], [51]. Typical strategies include solving relaxed subproblems, using mathematical programming inside a neighborhood search, or decomposing a large scheduling problem into smaller solver-based components. Compared with pure exact methods, matheuristics can reduce the computational burden and exploit problem structure more effectively. Nevertheless, their performance usually depends on a problem-specific decomposition, relaxation, neighborhood definition, or repair mechanism. For the problem studied here, designing such a matheuristic would require exposing the internal structure of the analytical plant model and developing dedicated subproblem formulations for machine assignment, manufacturing-way selection, sequencing, and shared-resource resolution. This would constitute a separate problem-specific solver

rather than a general baseline for evaluating the proposed migration mechanism.

Learning-based scheduling methods are gradually being applied in dynamic and complex scheduling environments. These methods learn a mapping from system states to scheduling decisions and can replace or enhance manually designed dispatching rules. Existing studies often formulate scheduling as a Markov decision process or a graph-based decision problem, and use deep reinforcement learning (DRL) [52], graph neural networks (GNNs) [53], or multi-agent reinforcement learning [54] to learn job selection, machine assignment, or sequencing policies. For example, DeepRM [55] shows that a scheduler can learn resource-allocation policies from experience, while GNN-based methods can exploit graph-structured representations of tasks and operations [56]. Deep multi-agent reinforcement learning has also been explored for decentralized scheduling decisions under dynamic environments [57]. Despite their potential, learning-based methods are not directly comparable to the proposed method without a substantial additional training and modeling pipeline. First, they usually require a large number of training instances or repeated interaction with a reliable simulator, and this training cost is separate from the final online decision time. When each evaluation is obtained from a plant-level analytical model, the cost of generating sufficiently diverse training trajectories can become substantial. Second, their performance can be sensitive to the training distribution. A policy trained on specific job topologies, machine configurations, manufacturing-way distributions, or resource-conflict patterns may degrade when these elements change. Third, deep models are usually black-box decision makers, which makes it difficult to diagnose scheduling failures, explain why a specific machine or manufacturing way is selected, or provide strict optimality or feasibility guarantees. These issues are particularly relevant in industrial scheduling settings, where robustness, interpretability, and constraint satisfaction are often as important as average solution quality.

Based on the above discussion, exact, matheuristic, and learning-based approaches are complementary to the proposed framework rather than direct substitutes for it. The problem considered in this paper is evaluated through an analytical plant model and contains heterogeneous discrete decisions, including machine assignment, manufacturing-way selection, and priority-based ordering under mutually exclusive shared manufacturing resources. Therefore, this paper focuses on improving a distributed evolutionary optimization framework for a black-box/grey-box multi-objective scheduling problem, rather than designing a new exact solver, a problem-specific matheuristic, or a trained scheduling policy. Within this framework, the key research question is how inter-island communication should be performed. The proposed linkage-based migration mechanism addresses this question by exploiting empirical variable dependencies during migration.

III. System Model and Problem Formulation

As with [5], [7], [20], this work focuses on the Extended Flexible Job-shop Scheduling problem (EFJSP). The EFJSP is a multi-objective optimization problem that can be found in most manufacturing factories, which aims at minimising the total makespan and monetary cost for producing a given amount of merchandise [20], [58]. It is an extension of the traditional Job-shop Scheduling Problem (JSP), in which each manufacturing task must be executed on a pre-assigned machine [59]. However, in an EFJSP, a task can be executed on a set of candidate machines, and the parallel tasks running on different machines will require mutually exclusively shared resources for processing. Compared to JSP, EFJSP better captures the realistic features of the manufacturing factories, however, it also significantly increases the complexity of the scheduling problem [33].

The considered EFJSP [7], [20] contains n tasks, $\tau_i \in \Gamma$, $i = 1, \ldots, n$, that need to be executed in a plant equipped with m machines, $\pi_j \in \Pi$, $i = 1, \ldots, m$ to produce the required goods. A machine π_j has two features: (i) the machine size δ_j and (ii) a set of manufacturing ways W_j to execute τ_i. Depending on δ_j, a machine π_j has a specific subset of tasks that it can execute. A manufacturing way $w_{j,k} \in W_j$ requires a specific shared resource that must be accessed in a mutual-exclusively fashion, i.e., two machines cannot execute simultaneously using the same manufacturing way k [20].

A task τ_i can be executed by a set of compatible machines in Π, denoted as $F(\tau_i) \subseteq \Pi$. The execution of τ_i on a compatible machine $\pi_j \in F(\tau_i)$ with a machine size δ_j and a manufacturing way $w_{j,k} \in W_j$ has an execution time $e^i_{j,k}$ and a monetary cost $c^i_{j,k}$. That is, a task can have various execution time and monetary costs, if being executed on different machines (featured by the machine size) with different manufacturing ways. Section VII presents a real-world example [5] illustrating the execution time and monetary cost of manufacturing a metal part using different machines and manufacturing ways.

For a given schedule $\mathcal{S}$, each task τ_i is assigned with (i) an *allocation* $\alpha_i = \{\pi_j, w_{j,k}\}$ defining the machine π_j (with a size of δ_j) and the manufacturing way $w_{j,k}$ of τ_i's execution; and (ii) a *unique priority* pri_i of τ_i. The allocation $\alpha_i = \{\pi_j, w_{j,k}\}$ determines the execution time and the monetary cost of τ_i, i.e., $e^i_{j,k}$ and $c^i_{j,k}$. The task priority pri_i defines the execution order of tasks on each machine, in which a task with a higher priority will be executed first [20]. The priority can be optimized so that parallel tasks (on different machines) with the same manufacturing way can be executed at different time slots, avoiding the additional blocking caused by the competition for the same shared resource [4].

With the allocation $\alpha_i = \{\pi_j, w_{j,k}\}$ and priority pri_i defined, the execution time $e^i_{j,k}$, the monetary cost $c^i_{j,k}$, the start time s_i, and a finish time f_i of each task τ_i in schedule $\mathcal{S}$ can be decided. First, the $e^i_{j,k}$ and $c^i_{j,k}$ are determined directly by $\alpha_i = \{\pi_j, w_{j,k}\}$. Then, the start time s_i can be obtained based on the priority ordering of tasks on each machine. Finally, the finish time f_i can be computed by $f_i = s_i + e^i_{j,k}$. Based on the above, a schedule $\mathcal{S}$ can be characterized by a makespan E (Equation (2)) and a total monetary cost C (Equation (3)), representing the time and financial cost for executing all the manufacturing tasks, respectively. The makespan is computed as the latest finish time among all tasks. The total monetary cost is the sum of the cost of executing all manufacturing tasks. For a given schedule, the E and C (along with f_i and $w^i_{k,j}$ of each task) are evaluated under the analytical plant model presented in [9].

$$E = \max_{\tau_i \in \Gamma} \Big\{ f_i \Big\} \tag{2}$$

$$C = \sum_{\tau_i \in \Gamma} \Big\{ c^i_{j,k} \mid \alpha_i = \{\pi_j, w_{k,j}\} \Big\} \tag{3}$$

The optimization problem of the targeted manufacturing scenario is formalised below. For a given set of manufacturing tasks Γ and machines Π, the objective of the considered problem is to find a schedule $\mathcal{S}$ such that the makespan E and monetary cost C can be minimised. For each task $\tau_i \in \Gamma$, the decision variables are (i) the machine and the manufacturing way $\alpha_i = \{\pi_j, w_{j,k}\}$ for execution and (ii) the priority pri_i. The value range of π_j is $F(\tau_i)$ (i.e., τ_i's compatible machines), the range of $w_{j,k}$ is W_j (i.e., the manufacturing ways of $\pi_j \in F(\tau_i)$), and the range of pri_i is [1, n].

$$\begin{array}{ll} \text{Given} & \Gamma,\ \Pi \\ \text{Goal} & \text{Minimise } E,\ C \\ \text{On} & \{\alpha_i = \{\pi_j, w_{j,k}\}, pri_i \mid \tau_i \in \Gamma\} \\ \text{s.t.} & \forall \tau_i \in \Gamma : \pi_j \in F(\tau_i), w_{k,j} \in W_j, pri_i \in [1, n] \\ & \forall \tau_a, \tau_b \in \Gamma, a \neq b : pri_a \neq pri_b \end{array} \tag{4}$$

As described above, the objective of the considered GA is to allocate and schedule the manufacturing tasks to machines so that

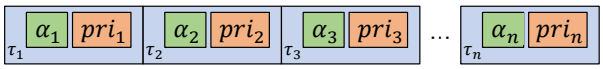


Fig. 2: The gene encoding of an individual with n tasks.

both E and C can be minimised. Accordingly, the encoding has to embrace both the spatial and temporal scheduling. Consequently, an individual contains genes of two types, as shown in Figure 2. Let ind denote the set of genes in a given individual. For n tasks (τ_1 to τ_n) that need to be scheduled, the number of genes is equal to $2 \times n$. The odd n genes (e.g. α_1 in the figure) indicates the target machine and the manufacturing way of each manufacturing task, i.e. $ind_{2x+1} \in \{\emptyset, \pi_1, \ldots, \pi_m\}$, where $x \in \{0, \ldots, n-1\}$. The remaining n genes (pri_i in the figure) specify the priorities of the n tasks, i.e. $ind_{2(x+1)} \in \mathbb{N}$, where $x = 0, \ldots, n-1$.

With the optimization problem and the gene encoding defined, a multi-objective GA can be applied to minimise both the makespan and the monetary cost, by producing the allocation and the priority of each manufacturing task. However, as described in the above sections, the traditional GA-based methods are becoming less effective due to the complexity of the target problem. In the following sections, a novel asynchronous island-based GA framework is proposed for the considered manufacturing optimization problem, in which each island is an independent GA instance that optimizes the target problem (i.e., Equation (4)) with asynchronous inter-island communication supported. Then, a fine-grained communication method is constructed, which only migrates the highly-dependent gene groups in one individual based on empirical linkage learning. Combining the above, we present a complete island-based GA framework that realises asynchronous GA islands with effective inter-island communication to improve the optimization quality.

IV. Asynchronous Island Model of the Cloud-based GA

This section presents the asynchronous island model for the proposed cloud-based GA. First, the asynchronous island model of the proposed GA framework is constructed in Section IV-A. Then, a series of individual-based migration methods are presented in Section IV-B, to improve the optimization performance of the proposed asynchronous IM-based GA. Finally, a cloud-based computing architecture is described in Section IV-C for the deployment of the proposed GA framework on the cloud.

A. Asynchronous Island-based GA Framework

In IM-based GAs, evolution is performed independently over a number of subpopulations (i.e. islands), where each island is optimized by an independent local GA instance. The classic IM follows a fully synchronous master-slave architecture: the iterations on all islands begin at the same time, triggered by a master node, and the iteration completion of all islands is synchronised with a barrier [4]. However, under this synchronous model, the early-finishing islands have to wait for the late-finishing ones in each iteration, which significantly undermines the effectiveness of the GA framework.

In this section, we propose an asynchronous island-based GA optimization framework which removes the need for frequent synchronisations but still allows the exchange of migrants. In the asynchronous island-based GA framework, each island maintains its own subpopulation. The GA engine in each island is executed asynchronously and does not communicate directly with each other. Instead, it uploads the non-dominated individuals it has found to a repository shared among all islands, implemented as a lightweight database (refer to as *DB* hereinafter). The non-dominated individuals stored in the *DB* by a given island can then be used by other islands to modify their current populations. By doing so, we enable the asynchronous communication between the GA islands. The optimization of an island stops after evolving a predefined number of generations.

Algorithm 1: Asynchronous island-based GA framework

inputs: I: number of iterations; P: number of individuals per island; S: number of stages; R: iteration number in a local optimum; QI: quality indicator; M: number of migration operations; DB: the central database that stores the PFA of all islands;

```
1  PFA = ∅, c = 0;
2  initialise a GA instance with P individuals;
3  for s=1,...,S do
4      execute the GA instance for I iterations;
5      add new non-dominated solutions (if any) into PFA;
6      if QI value of PFA obtained in stage (s) is not higher than
         that of stage (s-1) then
7          c = c + 1;
8          for m=1,...,M do
9              migrate an individual to the local population
                 (Algorithm 2 or Algorithm 3);
           end
10         if c == R then
11             c=0;
12             update the DB with the current PFA;
           end
       end
   end
13 update the DB with the final PFA;
```

Algorithm 1 describes the execution of an asynchronous island-based GA. The island is created with an empty local Pareto front approximation set (denoted as PFA) and an initial population with P individuals (lines 1-2). The island runs for S stages, and in each stage the GA instance executes for I iterations (lines 3-4). When a GA island finishes I iterations in each stage, it updates its PFA with new non-dominated solutions (if found, line 5). Then, a quality indicator is employed to evaluate the quality of the current PFA. If the quality of the PFA is not improved compared to the last stage (line 6), M migration operations are triggered, and each operation uses one individual of a remote PFA in the DB to modify the current population (line 8-9). In addition, if the quality of the PFA has not been improved for R iterations (i.e., has reached a local optimum), the current PFA of the island is updated to DB by overwriting the previous PFA of the same island stored in the DB (lines 10-12). When an island finishes execution, its PFA is updated to DB as the final result obtained by this GA island (line 13). Finally, when all islands are finished, the non-dominated individuals among all PFAs in the DB from all islands are reported as the Pareto front approximation of the target problem.

As described in Algorithm 1, an island can perform the *migration* operation, i.e., using the stored non-dominated individuals from another island to modify its current population at certain time points. In this work, we propose two approaches for migrating individuals between islands. The first method migrates complete individuals based on predefined selection and replacement operators (Section IV-B). The second method takes a deep examination of an individual to identify and migrate groups of highly dependent genes using empirical linkage learning techniques (Section V).

B. The Individual-based Migrations Between Islands

The individual-based migration is performed by a selection operator $select_op$ and a replacement operator $replace_op$. Algorithm 2 presents the working mechanism of the individual-based migration. For a local island that performs a migration operation, it randomly

Algorithm 2: The individual-based migration

inputs: DB: central database; POP: population of a local island; $select_op$: select an individual in a remote FPA in DB; $replace_op$: find an individual in the POP to replace;

1 ind_select = null; $ind_replace$ = null;
2 $remote_PFA$ = a randomly-chosen remote PFA in the DB;
3 **switch** *select_op* **do**
4 **case** *generic*:
5 ind_select = a newly-generated individual;
6 **case** *random*:
7 ind_select = a randomly-chosen individual in $remote_PFA$;
8 **case** *best*:
9 ind_select = the best individual from $remote_PFA$, based on the proximity to the ideal point $(0, 0)$;
10 **case** *diversity*:
11 ind_select = the most diverse individual from $remote_PFA$, based on the Crowding Distance;
 end
12 **switch** *replace_op* **do**
13 **case** *random replacement*:
14 $ind_replace$ = a randomly-chosen individual in POP;
15 **case** *worst replacement*:
16 $ind_replace$ = the worst individual in POP, based on the proximity to the ideal point $(0, 0)$;
 end
17 add ind_select to POP and remove $ind_replace$ in POP;

chooses a remote PFA in the DB (i.e., $remote_PFA$) as the source of the migration (line 2). The selection operator selects an individual from the $remote_PFA$ as the migrant based on a pre-defined selection strategy (lines 3-11). Then, the replacement operator identifies an individual to be removed in the current population (POP) of the local island using a pre-defined replacement strategy (lines 12-16). Finally, the migration is finished by replacing the local individual with the selected migrant. Below we describe the selection and replacement strategies in detail.

Four strategies for realising the selection operator are considered, as enumerated below. Note, as the optimization problem targeted in this work focuses on minimising both the makespan and monetary cost, the ideal point can be effectively set as $(0, 0)$ to allow the evaluation of an individual during migration.

- *Generic selection* does not perform the actual selection from the $remote_PFA$, but instead, it randomly generates a new individual as the migrant (lines 4-5). This strategy serves as the performance baseline for the remaining selection operators.
- *Random selection* randomly selects a solution from the $remote_PFA$ (lines 6-7).
- *Best selection* selects the best solution from the $remote_PFA$ (lines 8-9). The solution quality is evaluated by the proximity to the ideal point.
- *Diversity selection* selects the individual from $remote_PFA$ which has the highest diversity based on the Crowding Distance (CD) value [27]. The CD measures the average distance between the solution and its two closest neighbours (lines 10-11).

Then, to maintain a fixed size of the population in each island, a replacement operator must be applied during the migration. This paper considers the following two replacement strategies:

- *Random replacement* removes a randomly-selected solution in the POP of the current island (lines 13-14).
- *Worst replacement* removes the worst solution in the POP, in terms of the proximity to the ideal point (lines 15-16).

By combing the above selection and replacement strategies, we provide, in total, eight individual-based migration methods that can be pre-configured before the optimization process.

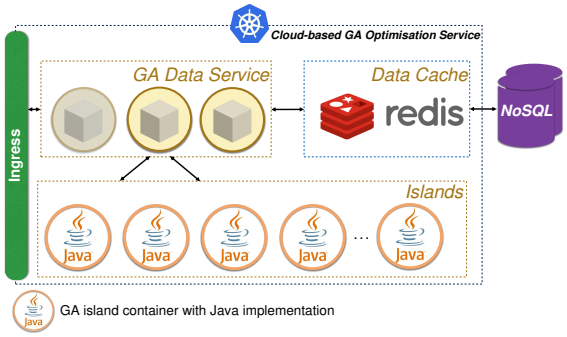


Fig. 3: The cloud deployment of the island model

C. Cloud Deployment

To deploy the above asynchronous island-based optimization framework in the cloud, an architecture is developed based on Docker and Kubernetes. This enables us to leverage the managed Kubernetes services from these cloud providers, rather than locally installed on-premises. By creating multiple instances of service components, Kubernetes automatically handles the load balancing and re-starts a faulty container once detected. This approach enables the proposed algorithm to be highly available during the operation.

The proposed distributed framework is not intended to replace lightweight single-machine optimization for very small instances. Instead, it targets larger-scale and repeated EFJSP optimization scenarios, where maintaining multiple GA islands as a reusable cloud service can provide scalable computational resources. In such a deployment, container startup is a one-time cold-start cost, and islands can be kept alive to serve multiple optimization requests. Therefore, the main runtime concern is the optimization process itself rather than repeatedly starting containers for each request.

The proposed architecture contains three major components, as shown in Figure 3. First, the `GA Data Service (data tier)` is responsible for the data communication between islands and storing the data in a persistent data storage. Second, the `Data Cache` is used to reduce the response time when the data service reads/writes data from/to the persistent storage. Third, the `GA Island` executes the proposed GA either on a managed cluster or on-premise.

The `GA Data Service` is the core component that handles data exchange between islands. It is highly available and automatically scales out/in according to the load of the requests from the GA islands. The `data cache` is designed to use a distributed key/value storage and is implemented by a Redis cluster, which supports both high availability and fast data exchange. The micro-service architecture employed by the proposed solution decouples the components so that the whole solution can be easily deployed to any distributed system. This enables the proposed solution to be provided as a cloud service by cloud providers and requires the minimum possible maintenance. The `GA Island` are implemented by Java, which communicates with the GA Data Service via REST API from within the cluster or outside of the cluster through the ingress.

All components follow the micro-service architecture which decouples the components by exposing their functionality using well-documented APIs following the RESTful [60] style. Consequently, the proposed solution can be easily provided as a cloud service and requires limited maintenance efforts.

V. Migration using Empirical Linkage Learning

As presented in Section II-C, the use of linkage learning may lead to reaching excellent results in multi-objective optimization [24],

Algorithm 3: Migration using Empirical Linkage Learning

inputs: ind: individual to discover the linkage from;
$destInds$: individuals to receive the migrated building blocks;

1 $linkScraps \leftarrow empty$;
2 $order \leftarrow$ GenerateRandomPermutation();
3 $weights \leftarrow$ GenerateRandomWeights();
4 $optInd \leftarrow$ OptGreedy($ind, weights, order$);
5 **for** $g = 1, ..., n$ **do**
6 $\quad distInd \leftarrow ind$;
7 $\quad distInd[g] \leftarrow$ randomMachineFor(g);
8 $\quad optDistInd \leftarrow$ OptGreedy($distInd, weights, order$);
9 $\quad linkScraps \leftarrow linkScraps+$ Diffr($optDistInd, optInd$);
end
10 $linkageTree \leftarrow$ CreateLT($linkScraps$);
11 **for** ***each*** $recInd$ *in* $destInds$ **do**
12 $\quad migrWeights \leftarrow$ GenerateRandomWeights();
13 $\quad recInd \leftarrow$ OptimalMixing($recInd, ind, linkageTree$);
end

[43]. Additionally, the recently proposed empirical linkage learning techniques (that never report the false linkage) support linkage of high quality that may be decisive for effectively solving many GA-hard problems [16], [17], [41]. Finally, migration operators that exchange problem decomposition models [29], or information obtained using these models [28] seem to be highly promising. Therefore, motivated by the above research, we propose an empirical linkage learning technique dedicated to the considered EFJSP problem. To the best of our knowledge, this is the first empirical linkage learning technique dedicated to multi-objective optimization. Based on this technique, we propose a novel migration operator using the empirical linkage learning, employed by an IM applied to solve the considered practical problem. The experiments show that the proposed migration method with empirical linkage learning significantly outperforms IMs using other migration operators and MOEA/D which is the EA dedicated to multi-objective optimization.

A. Empirical Linkage Learning and Migration based on Linkage

For the EFJSP problem considered in this work (see Section III, there often exist different groups of genes, each of which contains genes heavily dependent on each other, i.e., a value change of one gene affects the optimal value of another gene. For example, manufacturing tasks with different characteristics (e.g., size of the product) must be executed on different groups of machines. Thus, the scheduling and allocation decision of a given task can only affect tasks which share the same product size. In this context, we propose a finer-grained migration based on the empirical linkage learning technique. named the *linkage-based* migration. In contrast to the individual-based migrations, which always migrate a complete individual, this method examines and identifies genes with a high dependency in one individual, and only migrates such highly dependent genes to the destination island. The linkage-based migration is inspired by the 3LO [16] technique. Since 3LO was originally proposed for binary domains, we introduce the following three major modifications to adjust it to the optimization problem considered in this paper.

First, 3LO uses the First Improvement Hill Climber (FIHC) [16], [44] as a local optimizer. In binary search spaces, this is equivalent to choosing the gene value leading to the highest fitness. This statement is not necessarily true for non-binary discrete search spaces. Thus, instead of FIHC, we use the greedy optimization algorithm that considers all available gene values for the optimized gene and chooses the one that refers to the highest fitness.

TABLE I: An example of DSM generated from *linkage scraps*

Gene number	**1**	**2**	**3**	**4**
1	X	2	0	0
2	2	X	1	1
3	0	1	X	2
4	0	1	2	X

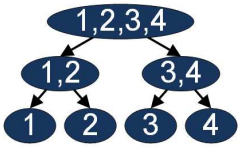


Fig. 4: The linkage tree for DSM from Table I

Second, the gene encoding of each individual is represented as $(\alpha_1, p_1, \alpha_2, p_2, ..., \alpha_n, p_n)$, where α_i is the machine the *i*th task is allocated to together with the machine manufacturing way and p_i influences the task order in the production plan on that machine. Although, recently, an empirical linkage learning technique was proposed for permutation-based problems [41], there is no 3LO version for such problems. Therefore, in the proposed linkage-based migration, we only discover linkage for genes that specify the designated machines and their modes (marked as α_i). The linkage discovery and migration of the priority-based part of the genotype will be addressed in future work.

Third, since EFJSP is multi-objective, for each 3LO procedure, we randomly choose a weight vector to transfer the problem into its single-objective counterpart temporarily for the duration of the 3LO linkage discovery procedure. Note that a similar mechanism of temporal multi-objective problem transformation into a single-objective problem was employed in MO-P3 [24] (see Section II-C).

The procedure of the proposed linkage-based migration is given in Algorithm 3. The entry information is an individual (ind) whose building blocks will be migrated and the set of individuals from the destination island that will receive the building blocks ($destInds$). First, 3LO is executed to discover the linkage from ind (lines 1-9). We generate a random optimization order (line 2) and weights (line 3) to make the problem single-objective during the linkage discovery procedure. Then, we disturb each gene in ind (line 7) and compare the optimized genotypes of ind and $distInd$ to obtain *linkage scraps* (lines 8-9). Then, we use the obtained *linkage scraps* to create a linkage tree (LT) (line 10). Finally, for each destination individual ($recInd$), we randomly choose the weight vector (line 12) and perform the optimal mixing (OM) [16], [44], [61]. For OM, we use the $migrWeights$ to make the problem single-objective, ind as a *donor* individual and $linkageTree$ to obtain linkage genes groups.

B. Running Example

To illustrate the working mechanism of the proposed linkage-based migration in detail, let us consider the following example. We examine a problem instance with 4 jobs and 3 machines. The donor individual is $ind = (2, 0.1, 1, 0.3, 3, 0.2, 1, 0.5)$, in which each odd gene refers to the machine (and its manufacturing way) that the job will be allocated on and each even gene indicates the order the task is inserted into the production plan on that machine. In the 3LO dedicated to the EFJSP problem, we consider only those genes that refer to machines (i.e., the odd genes). We assume that $optInd = (3, 0.1, 1, 0.3, 4, 0.2, 1, 0.5)$, in which the genotype of ind is greedily optimized in a random order (line 4 in Algorithm 2).

Now we disturb each allocation gene and obtain a *linkage scrap* for each disturbed gene (lines 5-9 in Algorithm 2). Let us assume that the disturbance of the first gene resulted in

TABLE II: Example configuration of a manufacturing part

Part	Machine size (MS)	Manufacturing Way (MW)	Cutting time (min)	Wire cost per part ($)	Machine cost per part ($)	Total cost per part ($)
	Small	1	2833.5	28.1	164.0	192.1
	Small	2	2956.2	28.1	140.3	168.4
	Small	3	3042.1	28.1	147.8	175.9
	Small	4	3174.1	28.1	136.8	167.0
$P1$	Medium	1	2033.5	30.2	242.9	273.1
	Medium	2	2156.2	30.2	208.4	238.6
	Medium	3	2242.1	30.2	196.1	237.1
	Medium	4	2674.1	30.2	189.0	230.0
	...	...	...	...	...	...

$distInd = (2, 0.1, 1, 0.3, 4, 0.2, 1, 0.5)$, and $optDistInd = (2, 0.1, \mathbf{2}, 0.3, \mathbf{1}, 0.2, \mathbf{2}, 0.5)$ (the different gene values in $optInd$ and $optDistInd$ are marked in bold). Thus, the *linkage scrap* discovered for such $optInd$ and $optDistInd$ will mark the fourth and sixth genes as being dependent and may be represented as $(*, *, 1, *, 1, *, 1, *)$, where $*$ and 1 refer to genes not included and included in the *linkage scrap*, respectively. Since 3LO dedicated to EFSPJ considers only the genes referring to machines, the linkage scrap reduced to these genes is equal to $(*, 1, 1, 1)$. Let us assume that for ind we have discovered the following set of *linkage scraps*: $(*, 1, 1, 1)$, $(1, 1, *, *)$, $(*, *, 1, 1)$, $(1, 1, *, *)$.

With gene scarps obtained, the DSM and the LT can be constructed (line 10 in Algorithm 2). For each gene pair contained in a single linkage scrap, we increase the appropriate DSM entry by one. For instance, for *linkage scrap* $(*, 1, 1, 1)$ we shall increase the entries that refer to the following gene pairs: (2,3), (2,4), and (3,4). Finally, the DSM that shall be created on the base of the collected scraps is presented in Table I. The strongest relations occur between genes (1,2) and (3,4). On the other hand, no direct relation was discovered between genes (1,3) and (1,4). For DSM presented in Table I, the corresponding LT is created by the following steps. First, the leaves referring to genes 2 and 3, and 1 and 4 are joined because they are the most dependent. Then, the nodes that contain pairs (2,3) and (3,4) will be joined, resulting in LT presented in Figure 4.

Finally, the nodes from LT will be used during the Optimal Mixing (OM), in which ind will be a *donor*, and the individuals from another island ($destInds$) will receive the information (line 11-13 in Algorithm 2). For each OM operation, we randomly choose a weight vector to make the problem single-objective. In OM, we use all LT nodes (except for the root) as crossover masks and insert genes from ind to $recInd$. There are three possible results for this insertion operation. First, if this operation improves fitness, we preserve the change and finish OM. Second, if the insertion changes an individual, but the fitness remains the same, we preserve the change and continue OM. Third, in case the insertion decreases the fitness, we revert the genotype change and continue OM. Let us consider the insertion example in which we try to improve $recInd = (1, 0.7, 2, 0.2, 3, 0.3, 1, 0.1)$, with the donor individual $ind = (3, 0.1, 1, 0.3, 4, 0.2, 1, 0.5)$ and the LT node groups genes (2,3). The modified genotype is $recInd = (1, 0.7, \mathbf{1}, 0.2, \mathbf{4}, 0.3, 1, 0.1)$, in which the modified genes are marked in bold and the linkage marks refer to only the allocation (odd) genes.

The version of 3LO proposed for the EFJSP problem enables 3LO linkage discovery for non-binary discrete multi-objective problems. Although our proposal ignores the priority part of the solution, the use of 3LO and the proposed migration operator can significantly improve the quality of the results, as shown in Section VI.

VI. Experimental Results

The evaluation is based on a real-world industrial EFJSP scenario rather than standard FJSP benchmarks suites. This choice is aligned with the goal of this work, which is to optimize a manufacturing problem with machine-size dependent compatibility, manufacturing-way selection, monetary cost, priority-based scheduling, and mutually exclusive shared manufacturing resources. These features are not directly available in standard FJSP benchmark suites. Therefore, the benchmark is scaled from the industrial instance to evaluate the proposed method under different problem sizes while preserving the structural characteristics of the target manufacturing scenario. This clarification also defines the evaluation scope of this work as the considered industrial EFJSP variant rather than generic FJSP benchmark suites.

Experimental Setup: A real-world manufacturing optimization problem [9] is employed for evaluation. The considered problem is a typical example of the EFJSP defined in Section III and can be found in discrete manufacturing factories [5]. The considered optimization problem consists of a set of 14 metal parts to be manufactured in a plant equipped with 3 Wire Electrical Discharge Machining (WEDM) machines of different machine sizes (MS), ranging from small to large. Each machine can operate in 4 manufacturing ways (MWs). Table II gives a configuration example of one manufacturing part, $P1$, which can be produced via 12 different approaches (i.e., combination of MSs with MWs) with various cutting time and monetary costs. A scaling factor $\kappa \in [1, 8]$ is introduced to control the size of the optimization problem. With $\kappa > 1$, the number of manufacturing parts and the number of machines are scaled by κ at the same time.

Unless stated otherwise, the original problem size ($\kappa = 1$) is applied and the optimization algorithm (Algorithm 1) is configured as the same with that in [4], [5], [7]: the number of islands is set to $NoI = 5$, the number of stages is $S = 40$, the population size of an island is $P = 50$, the number of iterations in a stage is $I = 20$, the threshold for triggering a DB update is $R = 3$, the number of migration operations is $M = 1$, and each island stops after executing S execution stages. For each experimental setting, 40 test cases are generated and each of them is evaluated under the considered methods described below. In each test case, the same initial population is ensured for each evaluated method to ensure the fairness of the evaluation.

Competing methods: The evaluation is conducted by comparing the following GA-based optimizers:

(i) MO-P3 proposed in [24];
(ii) Multi-Objective GOMEA (MO-GOMEA) in [43];
(iii) Gene Pattern Based Island Model (GePIM) proposed in [28];
(iv) NSGA-II in [62];
(v) the original MOEA/D proposed in [21];
(vi) the traditional island-based GA framework using MOEA/D as the local optimizer without migration between the islands (denoted as *No-Mig*);
(vii) the IM with eight individual-based migration variants presented in Section IV-B (denoted as *R+S* in which R indicates the replace operator and S gives the selection operator, e.g. *worst+diversity*);
(viii) the IM with the linkage-based migration proposed in Section V-A (denoted as *linkage*).

In addition, we note that for the NSGA-II and the original MOEA/D, their population sizes are set to the total population size of an IM-based method (i.e. $NoI \times P$). For each MOEA/D optimizer (including the IM-based ones), it is configured as with [7]: the neighbourhood size (i.e., the T value in [21]) is set to 30% of the population, the one-point crossover operator is applied, and the mutation rate is set to 10%.

Quality Indicator and Ranking: To evaluate the Pareto Front approximations produced by the evaluated methods, two multi-objective quality indicators (QI) are applied, namely the *Inverted Generational Distance* (IGD) [63], and the *Hypervolume* (HV) [64]. The IGD is

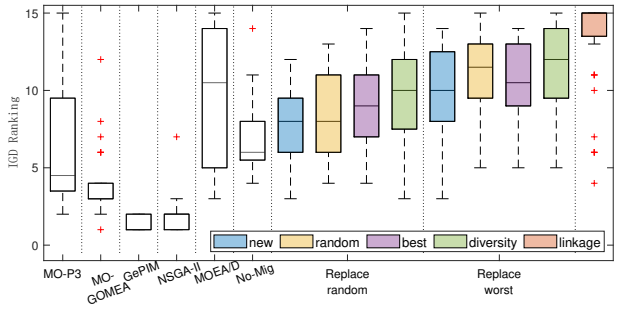


Fig. 5: IGD-based quality ranking of the evaluated methods.

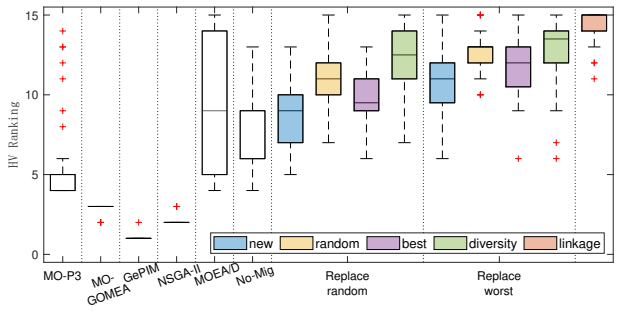


Fig. 6: HV-based quality ranking of the evaluated methods.

a frequently used QI that measures the most important aspects of the *PF*: its width and quality [65]. However, we acknowledge that the IGD is not Pareto-compliant [66]. Therefore, the HV is also applied to provide a more comprehensive evaluation, which is one of the most popular QIs that measure the space covered by a *PF* approximation [65]. It is worth noting that for the target problem, the reference PF or the ideal point is often infeasible to obtain in real-world scenarios. Thus, the non-dominated individuals among the solutions returned by all competing methods are set as the reference PF to allow the use of IGD, whereas the reference point is set to (0,0) (see Section IV-B) to enable the evaluation by HV.

In addition, to better illustrate the comparison under QIs with different value ranges, the ranking system used in [5], [7], [67] is applied in the evaluation, which reflects the quality of the *PF* approximations obtained by the evaluated methods under a given QI. For a set of evaluated methods in one test case, the methods are ordered by the quality of their *PF* approximations obtained by a given QI, and a method with a higher quality is assigned with a higher ranking value, i.e., from 1 to 15 for fifteen competing methods. As shown by the evaluation in [5], [7], [67], the rankings can be evaluated by the statistical analysis for pairwise comparisons with the Bonferroni correction taken into account.

A. Effectiveness of IM-based GA with Migrations

Figures 5 and 6 present the quality ranking of the optimization results obtained by the evaluated optimization methods, with IGD and HV applied, respectively. The individual-based migrations are grouped by the replacement methods (e.g. *replace worst* in the x-axis label) and the colors represent different selection operators (e.g. *diversity selection* in green as shown in the legend).

Observation: Below we present the major observations for each method, starting from the method with the lowest quality.

GePIM. GePIM obtains one of the lowest rankings under both IGD and HV, with a performance level close to that of NSGA-II. This observation indicates that directly applying linked-gene-group migration is not sufficient for the considered EFJSP. Although GePIM exchanges gene groups instead of complete individuals, its lightweight linkage construction is less effective in capturing the coupled dependencies induced by machine assignment, manufacturing-way selection, priority-based ordering, and mutually exclusive shared resources. As a result, the migrated gene groups may not consistently represent beneficial building blocks for the considered EFJSP.

TABLE III: The average Hamming Distance ranking of individuals in the final population.

MOEA/D	random + random	random + best	random + diversity	worst + random	worst + best	worst + diversity	linkage
1.3	6.9	4.9	9.2	4.5	3.1	6.9	4.8

NSGA-II. First, we observe that NSGA-II is significantly outperformed by MOEA/D and the IM-based variants. Similar results have been found in [7], [20].

MO-GOMEA. MO-GOMEA performs better than GePIM and NSGA-II, but it is still outperformed by the proposed linkage-based migration and the stronger IM-based variants. This is expected because MO-GOMEA constructs linkage models through statistical linkage learning and clusters the population according to different regions of the Pareto front. While this mechanism is effective for many multi-objective benchmark problems, it may become less reliable for the considered EFJSP, where the genotype combines allocation genes and priority-related genes and where the quality of a solution is affected by resource conflicts and scheduling interactions. In addition, clustering the population for different Pareto-front regions reduces the number of samples available for linkage estimation in each cluster, which may weaken the quality of the learned linkage model.

MO-P3. MO-P3 achieves rankings close to *No-Mig* under both IGD and HV. This suggests that the linkage diversity inherited from P3 helps maintain a certain level of population diversity, but it does not provide the same benefit as explicit inter-island migration in the proposed framework. Unlike the proposed method, MO-P3 does not exchange empirically discovered EFJSP-specific building blocks between islands through asynchronous migration. Therefore, MO-P3 can remain competitive with the island model without migration, but it fails to benefit from the cross-island propagation of problem-specific building blocks.

No-Mig. For the IM without migrations (i.e., the "No-Mig" variant), it demonstrates the second lowest effectiveness among all methods. Such an observation is expected because this method cannot exchange information between islands, which maintains and evolves on a number of separate subpopulations of a relatively small size, and hence, is more vulnerable to premature convergence. This observation highlights the necessity of effective migration strategies to realise inter-island communication in an IM-based GA framework.

Individual-based migration. For the replacement operators, we observe that the ones with the *replace random* are outperformed by those applying the *replace worst* under the same selection operator in a general case. This is expected with the underlying intuition that the *worst replacement* can improve the general quality of the population and facilitate the convergence by removing the worst individual [4], [7]. As for the selection operators, the *generic selection* (i.e., the baseline method) is outperformed by others as it simply generates a new individual without any optimization, which is likely to be dominated by the existing individuals in the population.

In addition, we observe the *best selection* is outperformed by *random selection* and *diversity selection*, under the same replacement operator. The reason is that always migrating the best solution from another island (which is likely to be the same in several iterations) would undermine the diversity of the entire population, hence, resulting in a low optimization quality. This is justified by

TABLE IV: The *Average* and *Median* ranking under IGD and HV of the optimization approaches.

$\kappa = 1$	*Rank*	MO-P3	MO-GOMEA	GePIM ...	NSGA-II	MOEA/D	*No-Mig*	random + new	random + random	random + best	random + diversity	worst + new	worst + random	worst + best	worst + diversity	*linkage*
Avg.	IGD	6.3	3.775	1.45	1.85	9.575	6.875	7.825	8.25	8.7	9.65	10.25	10.925	10.4	**11.475**	**13.3**
	HV	5.45	2.925	1.025	2.075	9.25	7.225	9.025	10.925	9.850	12.125	10.95	12.65	11.525	**12.925**	**14.425**
Med.	IGD	4.5	3.0	1.0	2.0	10.5	6.0	8.0	8.0	9.0	10.0	10.0	11.5	10.5	**12.0**	**15.0**
	HV	4.0	3.0	1.0	2.0	9.0	6.0	9.0	11.0	9.5	12.5	11.0	13.0	12.0	**13.5**	**15.0**

TABLE V: The p-values for the null-hypothesis that the quality of results of the linkage-based migration is the same as competing methods.

NSGA-II	MOEA/D	*No-Mig*	random + new	random + random	random + best	random + diversity	worst + new	worst + random	worst + best	worst + diversity
5.5×10^{-17}	2.0×10^{-9}	1.3×10^{-14}	8.4×10^{-14}	4.5×10^{-9}	4.5×10^{-13}	4.2×10^{-8}	2.7×10^{-13}	1.1×10^{-6}	3.8×10^{-11}	5.4×10^{-4}

Table III, which reports the Hamming Distance [68] of the genotype of individuals in the final population of each method. As shown in the table, the methods with the *best selection* operator demonstrate the lowest Hamming Distance. Instead, the *random selection* and *diversity selection* can improve the population diversity, hence, are more beneficial for the optimization performance.

MOEA/D. As shown in the figure, the MOEA/D maintains a population of a large size. Thus, it is less likely to converge prematurely. Therefore, its results are competitive to the results of IMs using individual-based migrations (see Figures 5 and 6). However, we observe that under the HV indicator (Figure 6), the quality of MOEA/D has the largest variation among all methods. This indicates the quality of its final results largely depends on the initial population. In contrast, the IM-based GA with migrations can produce more predictable results. Furthermore, we observe that the method *worst+diversity* outperforms MOEA/D under the HV. In addition, compared to the IM-based methods, the original MOEA/D requires significantly higher computation cost, due to the maintenance of a large population size (see Table VII later in Section VI-C). In contrast, in the proposed IM-based methods, each parallel optimizer is applied on a small size of population and can communicate using asynchronous migration. By doing so, the IM-based methods can (i) effectively reduce the computation cost of the optimization process and (ii) provide competitive results through effective inter-island communication using migrations. As shown in Table VII with $\kappa = 1$, to deliver the above results, the computation time of MOEA/D is 3.78x and 2.70x higher than that of the *worst+diversity* migration strategy and the linkage-based migration, respectively.

Linkage-based migration. Finally, we observe that linkage-based migration significantly outperforms all competing methods under both quality indicators. This observation justifies the intuition that only migrating the highly-relevant building blocks identified by empirical linkage learning is more beneficial to the final optimization results for two reasons. First, we exchange the important information reduced to a building block (the beneficial values on certain inter-dependent genotype positions) effectively. Second, thanks to the direct exchange of building blocks, we omit migrating the whole individual, which prevents the premature convergence (the single high-rated building block is less likely to dominate the subpopulation than the whole individual) while maintaining a good level of population diversity. Therefore, the proposed linkage-based migration method can lead to the optimization results of a higher quality.

Observation: Table IV summarises the average and median ranking of all competing methods reported in Figures 5 and 6. From the table we can first observe that the IM not employing any migration is outperformed by all other methods. Among IMs using the individual-based migrations, IM using the *replace worst* and *diversity selection* operators (i.e., *worst+diversity*) is the most effective. It also outperforms MOEA/D under the HV indicator. In addition, we can observe that the IM with linkage-based migration outperforms all the competing methods under both quality indicators. This again justifies the effectiveness of the proposed linkage-based migration method.

TABLE VI: The average IGD and HV quality ranking with scaled computation time (ind.-mig. — the most competitive individual-based method observed, i.e., the *worst+diversity*).

Time	IGD				HV			
	MO-P3	NSGA-II	ind.-mig.	*linkage*	MO-P3	NSGA-II	ind.-mig.	*linkage*
10s	6.2	1.9	11.0	**13.2**	5.5	2.1	13.2	**13.9**
20s	6.1	1.9	11.0	**13.1**	5.3	2.1	13.1	**14.3**
30s	6.2	1.9	10.9	**14.0**	5.4	2.1	12.8	**14.4**
40s	6.0	1.9	10.9	**14.4**	5.4	2.1	12.6	**14.4**
50s	5.8	1.9	11.1	**14.2**	5.3	2.1	12.5	**14.6**
60s	6.0	1.9	11.2	**14.0**	5.2	2.0	12.5	**14.5**
70s	5.9	1.9	10.7	**14.3**	5.4	2.0	12.1	**14.5**
80s	5.9	1.9	10.9	**14.1**	5.4	2.0	11.7	**14.6**
90s	6.0	1.9	10.2	**14.1**	5.2	2.1	12.0	**14.6**
100s	5.7	1.9	10.4	**14.4**	5.2	2.0	11.9	**14.4**

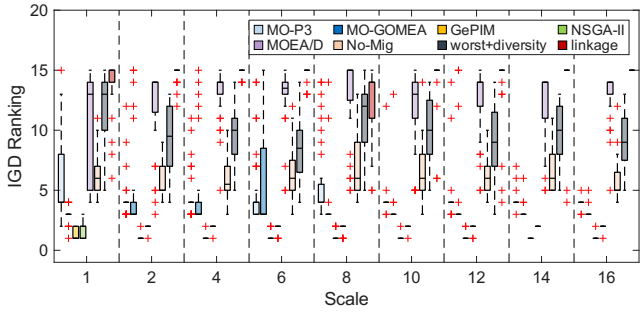


Fig. 7: IGD-based quality ranking across different scales.

To demonstrate the statistical significance of the improvements obtained by the linkage-based migration, Table V shows the p-values of the comparison between the linkage-based migration and each of the other evaluated methods by the Wilcoxon signed rank test. The null hypothesis is that *there are no statistically significant differences in the resulting quality between the compared methods*. The results of the statistical tests (with the Bonferroni correction taken into account) with a 5% significance level confirm that the IM with linkage-based migration outperforms the competing methods significantly. In addition, we note that the obtained p-values are low, and can confirm the supremacy of the linkage-based strategy also for lower significance levels, e.g., 0.05%.

Observation: As described above, the linkage-based migration performs best among all methods. However, in contrast to individual-based migrations, the linkage-based migration requires a search algorithm to construct the inter-dependencies of genes, i.e., the linkage tree, which is more computationally extensive. To further justify the effectiveness of this method, an experiment is conducted to compare the MOEA/D, the best individual-based migration strategy observed (i.e., *worst+diversity*, denoted as ind.-mig. for short), and the linkage-based migration, under the same computation time. Table VI presents

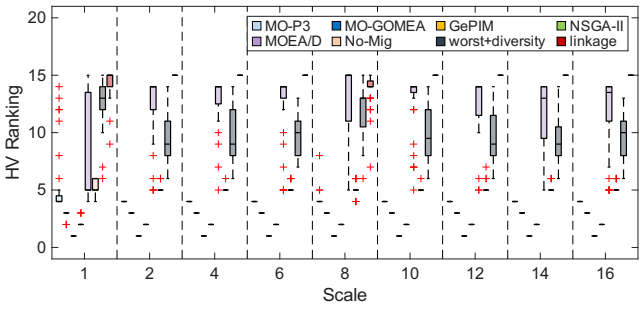


Fig. 8: HV-based quality ranking across different scales.

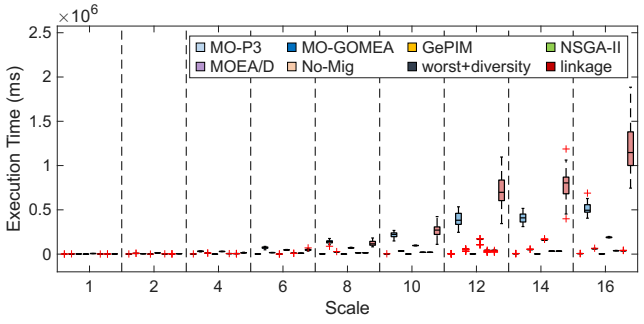


Fig. 9: The execution time across different scales.

the average quality ranking of both IGD and HV for the above methods under scaled computation time. From this table, we can observe that given the same computation time, the linkage-based method still consistently outperforms all the competing methods while the MOEA/D demonstrates the lowest quality. This observation is also confirmed by the Wilcoxon signed-rank test at a significance level of 5%, with the Bonferroni correction taken into account.

B. Scalability of the Proposed IM-based GA Framework

Figures 7, 8, and 9 report the quality ranking and execution time of MO-P3, MO-GOMEA, GePIM, NSGA-II, MOEA/D, No-Mig, the best individual-based migration variant (*worst+diversity*), and the linkage-based migration as the problem size (κ) increases.

Observation: The above figures show that the linkage-based migration achieves consistently better IGD and HV rankings than MO-P3, MO-GOMEA, GePIM, NSGA-II, MOEA/D, No-Mig, and *worst+diversity* across all considered problem sizes. This advantage can be attributed to the linkage learning mechanism, which explicitly captures the dependencies among decision variables and migrates correlated variables jointly. Therefore, useful variable interactions are better preserved during migration, while purely diversity-driven migration may disrupt such interactions and provide less effective search guidance. The execution-time results in Figure 9 further show that this improvement is obtained with additional computational overhead, especially for larger problem sizes, because linkage identification and linkage-guided migration introduce extra search cost.

Another observation is that MOEA/D outperforms *worst+diversity* under all problem sizes with $\kappa \geq 2$, indicating that decomposition-based search provides more stable guidance than the heuristic *worst+diversity* strategy when the problem size increases. These observations are consistent with the previous quality-comparison results, where linkage-based migration achieved the best overall IGD/HV performance and MOEA/D remained stronger than *worst+diversity* for $\kappa \geq 2$. In addition, we can also observe that GePIM and NSGA-II yield the worst performance among all evaluated methods, which fail to explicitly capture the coupling dependencies induced by machine

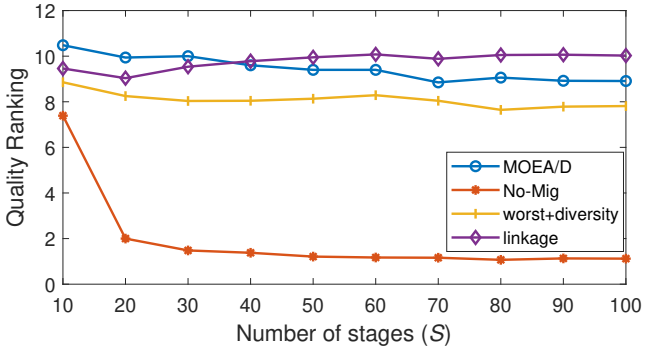


Fig. 10: Average quality ranking with increasing number of stages S when $\kappa = 2$.

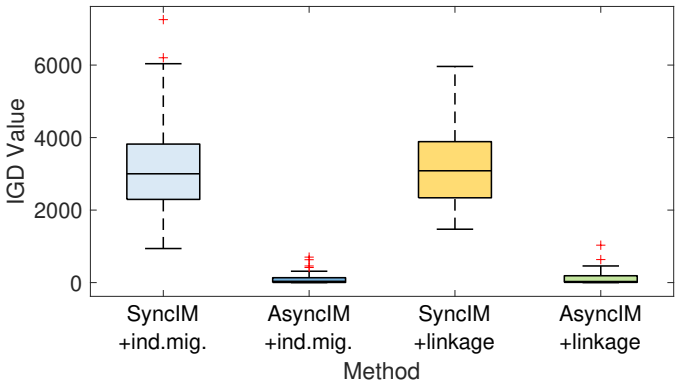


Fig. 11: The IGD comparison of synchronous and asynchronous IMs.

allocation, manufacturing method selection, priority-based sorting, and mutually exclusive resource sharing.

An interesting observation can be made for the linkage-based migration and MOEA/D for $\kappa = 2$. In this case, the results by these methods are similar while for $\kappa \neq 2$, the linkage-based migration significantly outperforms MOEA/D. The explanation for this observation is that for $\kappa = 2$, the MOEA/D can converge fast enough to compete with the linkage-based migration. To verify this hypothesis, for $\kappa = 2$ we increase the number of stages from $S = 10$ to $S = 100$ and report the resulting quality obtained at different stages under HV. We consider four representative methods: the original MOEA/D, *No-Mig*, the *worst+diversity* migration, and linkage-based migration. For MOEA/D, an increment on S indicates an increase of 20 in the number of iterations. The results reported in Figure 10 show that, indeed, at the early computational stages MOEA/D converges faster than the linkage-based migration. Then, after 40 stages (to recall, the computation budget for the main experiments is 40 stages), the MOEA/D and the linkage-based migration propose results of similar quality. Finally, the linkage-based migration outperforms MOEA/D for $S > 40$.

The above observations support the explanation for the results of the MOEA/D and linkage-based migration in Figure 8, where the MOEA/D has a faster convergence, and after which the algorithm is getting stuck. On the other hand, although the convergence of linkage-based migration is slower, this method can continuously improve the results for a larger number of iterations. In effect, when the computation budget increases, the linkage-based migration significantly outperforms the MOEA/D.

To isolate the contribution of asynchronous execution, we further compare the proposed asynchronous IM with its synchronous counterpart under the same migration operators. Two representative migration strategies are considered: the best individual-based

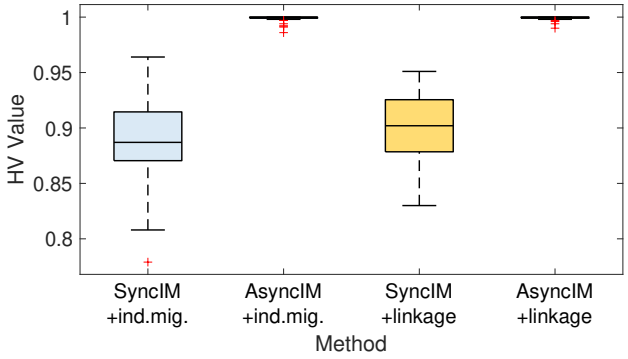


Fig. 12: The HV comparison of synchronous and asynchronous IMs.

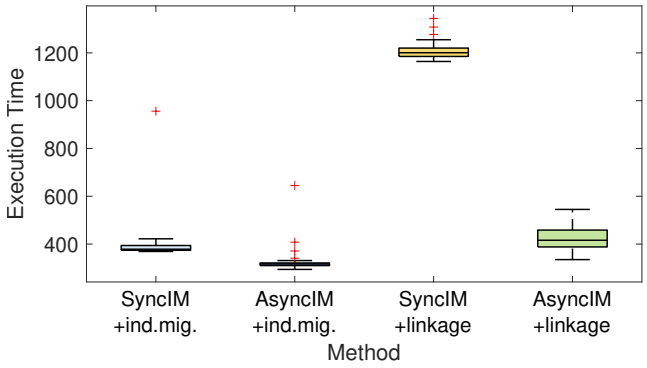


Fig. 13: The ET comparison of synchronous and asynchronous IMs.

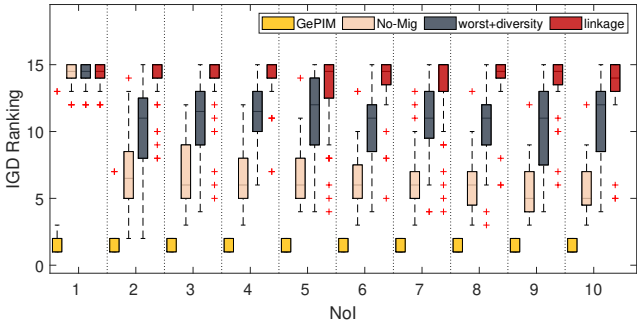


Fig. 14: IGD-based quality ranking across different NoI.

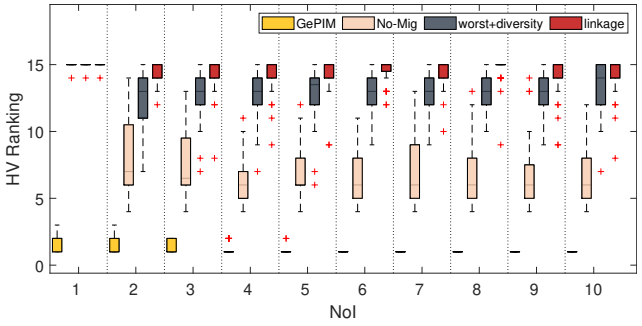


Fig. 15: HV-based quality ranking across different NoI.

migration strategy, i.e., ind.-mig., and the proposed linkage-based migration. In total, four methods are compared: Sync-IM+ind.-mig., Async-IM+ind.-mig., Sync-IM+linkage, and Async-IM+linkage. All methods are evaluated under the same experimental setup e.g., population size, number of islands and number of stages.

Observation: Figures 11, 12, and 13 report the IGD value, HV value, and execution time of the compared methods. The results show that, under the same migration operator, the asynchronous variants achieve better optimization quality than their synchronous counterparts. Meanwhile, asynchronous execution generally reduces execution time. This is because, in the synchronous island model, migration is performed at synchronization points, and faster islands may have to wait for the slowest island before exchanging migrants and proceeding to the next stage. In contrast, the asynchronous island model does not require such global synchronization; each island can continue its search and perform migration when the migration condition is reached. This reduces idle waiting time and allows useful information to be propagated more promptly across islands. Moreover, it seems that Async-IM with ind.-mig. and Async-IM with linkage-based migration achieve comparable IGD and HV values in this experiment. This is because the synchronous variants produce much larger IGD values and lower HV values, the visual scale of the box plots is dominated by the sync–async gap, which makes the finer difference between the two asynchronous migration operators less visually pronounced. Nevertheless, linkage-based migration still provides better optimization quality than ind.-mig. under the asynchronous setting, because it transfers empirically discovered EFJSP-specific building blocks rather than complete individuals. This suggests that asynchronous execution mainly reduces synchronization overhead and improves the timeliness of inter-island information exchange, while linkage-based migration further improves the quality of the exchanged information. The additional execution time of linkage-based migration is caused by empirical dependency discovery. These results confirm that the benefit of asynchronous execution mainly lies in reducing synchronization overhead. In contrast, the difference between ind.-mig. and linkage-based migration reflects the quality-runtime trade-off of different migration mechanisms.

Sensitivity to the number of islands. To evaluate the sensitivity of the proposed framework to the number of islands, we compared the performance of representative methods across a range of NoI values from 1 to 10. For a fair comparison, the total optimization budget and other algorithmic settings are kept identical across different NoI values.

Observation: Figures 14 and 15 report the IGD and HV rankings under different NoI values. The linkage-based migration consistently achieves high rankings across the tested island numbers, indicating that its advantage is not tied to a specific NoI configuration. The *worst+diversity* migration also shows relatively stable performance, which confirms the robustness of the individual-based migration strategy. In contrast, GePIM obtains relatively low rankings in most settings. This is mainly because its linked-gene-group migration relies on a lightweight linkage identification mechanism, which may not accurately capture the complex gene dependencies in the considered EFJSP. In this problem, the effect of each gene is jointly influenced by machine allocation, manufacturing-way selection, task priority ordering, and shared-resource constraints. Therefore, the linked gene groups identified by GePIM may provide limited useful building blocks for inter-island migration, leading to inferior optimization quality.

The results also show an interesting trend for *No-Mig*. When $NoI = 1$, *No-Mig* degenerates to a single-population optimizer, where all individuals evolve together and useful genetic material can be propagated within the whole population. Therefore, it performs competitively in this setting. However, when NoI increases, the population is divided into several smaller isolated islands. Since *No-Mig* disables migration, these islands cannot exchange high-quality individuals or useful building blocks. As a result, each island has reduced search coverage and becomes more likely to converge

prematurely, which explains the sharp performance degradation of *No-Mig* when $NoI > 1$. This observation indicates that the benefit of the island model does not come from simply increasing the number of islands. Effective inter-island communication is necessary to exploit multiple islands.

Increasing NoI does not monotonically improve the optimization quality. A larger number of islands may increase population diversity and communication opportunities, but it can also reduce the effective search capability of each island and introduce additional migration-related overhead. Therefore, the number of islands should be selected by considering both optimization quality and execution cost. The results justify the use of $NoI = 5$ in the main experiments and show that the proposed linkage-based migration remains effective under different island-model configurations.

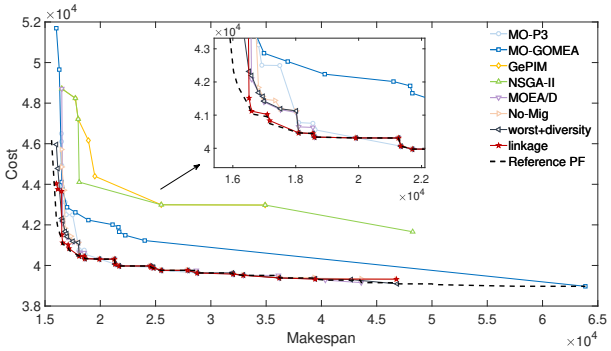


Fig. 16: Makespan-cost Pareto-front approximations of representative methods and the reference Pareto front.

Pareto Front Construction and Visualization. The reference Pareto front is constructed as the non-dominated union of all solutions obtained from all evaluated methods over all independent runs, including MO-P3, MO-GOMEA, GePIM, NSGA-II, MOEA/D, *No-Mig*, all individual-based migration variants, and the proposed linkage-based migration. This ensures the comparison is performed against the best approximation available from the complete set of evaluated algorithms.

Observation: The proposed linkage-based migration still achieves the best IGD performance among all evaluated methods. To further illustrate the quality differences beyond the indicator-based box plots, Figure 16 visualizes the obtained Pareto fronts in the makespan-cost objective space for representative methods, including MO-P3, MO-GOMEA, MOEA/D, *worst+diversity*, and the proposed linkage-based migration. The figure shows the trade-off curves between makespan and monetary cost, while the zoomed-in region highlights the differences among competing methods. From the results, the proposed linkage-based migration produces solutions closer to the lower-left region of the objective space, indicating better makespan-cost trade-offs. Compared with MOEA/D and the best individual-based migration variant, the proposed method improves the Pareto-front approximation over a range of trade-off preferences rather than a single objective.

C. Communication and Computation Overhead

Finally, we report the runtime communication, computation and monetary cost of the proposed IM-based GA framework.

Observation: Figure 17 presents the numbers of DB updates and migrations issued during the optimization process, under each evaluated method. The importance of this metric stems from the cloud communication overhead, as discussed later in this section, and the effectiveness of the communications of each migration strategy. As shown on the left-hand side of the figure, the *new selection*

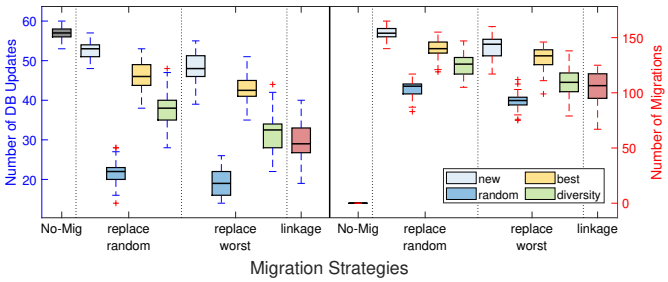


Fig. 17: The number of DB updates (10 leftmost boxes) and migrations (9 rightmost boxes) made by each migration method.

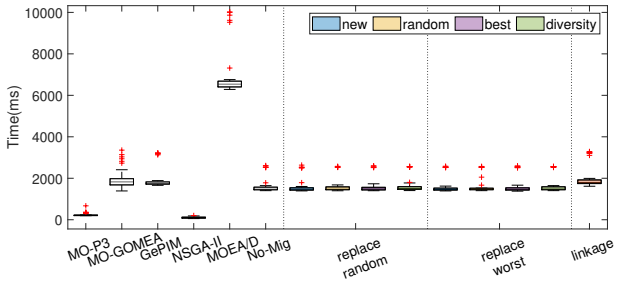


Fig. 18: Runtime of the evaluated methods.

and *best selection* operators have led to a larger number of DB updates and migrations than other selection operators, under both remove operations. This indicates the islands under these operations are more likely to get stuck at local optima but the migrations are less prone to escape from the local optima. In contrast, the *diversity selection* and the linkage-based migration have yielded fewer communication requests than the above-enumerated operators. This indicates their islands are less likely to get stuck at a local optimal, and communications between islands can effectively boost the optimization process.

Figure 18 reports the wall-clock time of all evaluated methods, including the standalone baselines and the proposed IM-based variants. We observe that MOEA/D requires more wall-clock time than the other methods. This overhead is not primarily due to a larger population size, as all standalone baselines are configured with comparable sizes. Instead, it stems from MOEA/D's decomposition-based neighborhood update mechanism, where each candidate solution necessitates scalarised fitness calculations and update checks across multiple neighboring subproblems. For the considered EFJSP, each candidate evaluation is computationally expensive because the genotype must be decoded into machine allocations, manufacturing-way selections, task priorities, and a feasible schedule under shared-resource constraints. Therefore, these additional update operations lead to higher computation time than NSGA-II, MO-P3, and MO-GOMEA. In contrast, the proposed IM-based methods optimize smaller island populations and exchange information asynchronously. The results also show that individual-based migrations have runtime close to *No-Mig*, whereas the proposed linkage-based migration incurs higher overhead due to empirical linkage discovery and optimal mixing.

Observation: Below we report the computation and monetary cost for executing the proposed asynchronous IM-based GA framework on the Amazon EKS cloud. The typical Amazon EC2 instances with ECUs ranging from 13 to 68 were used as worker nodes in the Amazon EKS cluster, where an ECU is a computing power of a physical server with a 1.0-1.2GHz CPU. Table VII presents the computation cost of the traditional MOEA/D, the individual-based

TABLE VII: Average computation cost (in *ms*) of the GA frameworks

($\kappa =$)	1	2	3	4	5	6	7	8
MOEA/D	61,564	73,416	120,064	186,440	241,204	366,448	560,544	639,296
ind.-mig.	16,280	18,400	30,384	45,408	62,008	90,228	111,404	152,976
linkage	22,800	30,308	61,920	89,492	174,556	358,812	491,688	560,544

migration (using *worst+diversity* as example) and the linkage-based migration with $\kappa \in [1, 8]$. As shown in the table, the original MOEA/D has the highest computation cost due to maintaining a large population size, which is about 3.78x and 2.70x higher than the individual-based and linkage-based migrations with $\kappa = 1$, respectively. With the scale factor $\kappa = 8$, the execution of the algorithm has taken about 639.3s with the original MOEA/D algorithm, which is 4.19x and 1.14x compared to that of the individual-based and linkage-based migration, respectively. The above observations show that the individual-based migrations have the lowest computation cost and are more scalable than other evaluated methods in terms of computational cost. In addition, the linkage-based migration outperforms the original MOEA/D in terms of both better optimization results and lower runtime overhead for all experimental settings.

In addition, it is worth reporting that the total monetary cost of executing the algorithm once with any migration strategies using EC2 (including both the computation time and the data transfer) was lower than 10 USD. Additionally, 0.10 USD was charged for an Amazon EKS cluster hourly usage in October 2021. Please refer to https://aws.amazon.com/eks/pricing/ for the cost of using Amazon EKS. As informed by the company formulating the real-world problem addressed in this paper, these costs constitute only 0.02 percent of the total production cost of the considered parts.

VII. Conclusion

This paper proposes a novel cloud-based optimisation framework based on the asynchronous island model. Each island represents a subset of the population that is evolved locally by a multi-objective genetic algorithm optimiser. Aiming at preserving population diversity and improving optimisation performance, nine migration strategies are proposed, including eight individual-based migration methods that migrate complete individuals and a linkage learning-based migration method that only migrates certain genes of the selected individual. All proposed methods are implemented following the micro-service architectural style and are deployed to a Kubernetes cluster in Amazon EKS. The proposed methods are evaluated under a real-world problem from a smart factory representing the branch. In future work, we will investigate migration topologies different from the fully connected graph used in this paper, e.g., a ring structure. We will also extend the linkage learning technique to support the permutation-based genes (e.g. priority of manufacturing tasks) and the optimisation process of each island.

References

[1] J. Wan, S. Tang, D. Li, M. Imran, C. Zhang, C. Liu, and Z. Pang, "Reconfigurable smart factory for drug packing in healthcare industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 507–516, Jan 2019.

[2] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, "Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination," *Computer networks*, 2016.

[3] L. Zhang and T. Wong, "Distributed genetic algorithm for integrated process planning and scheduling based on multi agent system," *IFAC Proceedings Volumes*, vol. 46, no. 9, pp. 760 – 765, 2013, 7th IFAC Conference on Manufacturing Modelling, Management, and Control. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016343798

[4] S. Zhao, P. Dziurzanski, M. Przewozniczek, M. Komarnicki, and L. S. Indrusiak, "Cloud-based dynamic distributed optimisation of integrated process planning and scheduling in smart factories," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '19. New York, NY, USA: ACM, 2019.

[5] P. Dziurzanski, S. Zhao, M. Przewozniczek, M. Komarnicki, and L. S. Indrusiak, "Scalable distributed evolutionary algorithm orchestration using docker containers," *Journal of Computational Science*, 2020.

[6] Z.-H. Zhan, X.-F. Liu, H. Zhang, Z. Yu, J. Weng, Y. Li, T. Gu, and J. Zhang, "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 704–716, 2016.

[7] S. Zhao, H. Mei, P. Dziurzanski, M. Przewozniczek, and L. S. Indrusiak, "Cloud-based integrated process planning and scheduling optimisation via asynchronous islands," in *International Conference on the Economics of Grids, Clouds, Systems, and Services*. Springer, 2019, pp. 247–259.

[8] P. Dziurzanski, S. Zhao, S. Scholze, A. Zilverberg, K. Krone, and L. S. Indrusiak, "Process planning and scheduling optimisation with alternative recipes," *at-Automatisierungstechnik*, vol. 68, no. 2, pp. 140–147, 2020.

[9] P. Dziurzanski, J. Swan, L. S. Indrusiak, and J. Ramos, "Implementing digital twins of smart factories with interval algebra," in *2019 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2019, pp. 941–948.

[10] Z. Skolicki, *Linkage in Island Models*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 41–60.

[11] X. Zhang, Z.-H. Zhan, W. Fang, P. Qian, and J. Zhang, "Multipopulation ant colony system with knowledge-based local searches for multiobjective supply chain configuration," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 3, pp. 512–526, 2022.

[12] S. Di Martino, F. Ferrucci, V. Maggio, and F. Sarro, "Towards migrating genetic algorithms for test data generation to the cloud," in *Software Testing in the Cloud: Perspectives on an Emerging Discipline*. IGI Global, 2013, pp. 113–135.

[13] N. Ma, X.-F. Liu, Z.-H. Zhan, J.-H. Zhong, and J. Zhang, "Load balance aware distributed differential evolution for computationally expensive optimization problems," in *GECCO Proceedings Companion, 2017*. ACM, 2017, pp. 209–210.

[14] D. Whitley, "Next generation genetic algorithms: a user's guide and tutorial," *Handbook of metaheuristics*, pp. 245–274, 2019.

[15] M. W. Przewozniczek, B. Frej, and M. M. Komarnicki, "On measuring and improving the quality of linkage learning in modern evolutionary algorithms applied to solve partially additively separable problems," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, ser. GECCO '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 742–750.

[16] M. W. Przewozniczek and M. M. Komarnicki, "Empirical linkage learning," *IEEE Transactions on Evolutionary Computation*, 2020.

[17] M. W. Przewozniczek, M. M. Komarnicki, and B. Frej, "Direct linkage discovery with empirical linkage learning," in *GECCO*, 2021.

[18] S. Zhao, H. Mei, P. Dziurzanski, M. Przewozniczek, and L. S. Indrusiak, "Cloud-based integrated process planning and scheduling optimisation via asynchronous islands," in *Economics of Grids, Clouds, Systems, and Services*, K. Djemame, J. Altmann, J. Á. Bañares, O. Agmon Ben-Yehuda, and M. Naldi, Eds. Cham: Springer International Publishing, 2019, pp. 247–259.

[19] D. E. Goldberg, "The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity," *Evolutionary Design by Computers*, pp. 105–118, 1999.

[20] P. Dziurzanski, S. Zhao, J. Swan, L. S. Indrusiak, S. Scholze, and K. Krone, "Solving the multi-objective flexible job-shop scheduling problem with alternative recipes for a chemical production process," in *Applications of Evolutionary Computation*. Springer International Publishing, 2019.

[21] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE on Evolutionary Computation*, 2007.

[22] J. Turek, J. L. Wolf, and P. S. Yu, "Approximate algorithms scheduling parallelizable tasks," in *Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures*, 1992, pp. 323–332.

[23] A. Antoniadis, M. Eliáš, A. Polak, and M. Venzin, "Approximation algorithms for combinatorial optimization with predictions," *arXiv preprint arXiv:2411.16600*, 2024.

[24] M. W. Przewozniczek, P. Dziurzanski, S. Zhao, and L. S. Indrusiak, "Multi-objective parameter-less population pyramid for solving industrial process planning problems," *Swarm and Evolutionary Computation*, vol. 60, p. 100773, 2021.

[25] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12.

[26] K. Deb and et al., "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. on Evolutionary Computation*, 2002.

[27] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," in *International conference on parallel problem solving from nature*. Springer, 2000, pp. 849–858.

[28] M. Komarnicki and M. W. Przewozniczek, "Linked genes migration in island models," in *Proceedings of the 8th International Joint Conference on Computational Intelligence (ECTA 2016)*, vol. 3, 2016, pp. 30–40.

[29] L. delaOssa, J. A. Gámez, and J. M. Puerta, "Migration of probability models instead of individuals: An alternative when applying the island model to edas," in *Parallel Problem Solving from Nature*, 2004.

[30] N. Melab, M. Mezmaz, and E. . Talbi, "Parallel hybrid multi-objective island model in peer-to-peer environment," in *19th IEEE International Parallel and Distributed Processing Symposium*, 2005.

[31] P. Salza, F. Ferrucci, and F. Sarro, "Develop, deploy and execute parallel genetic algorithms in the cloud," in *GECCO*, 2016.

[32] S. Lemaignan, A. Siadat, J. . Dantan, and A. Semenenko, "Mason: A proposal for an ontology of manufacturing domain," in *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*, June 2006, pp. 195–200.

[33] P. Dziurzanski, J. Swan, and L. S. Indrusiak, "Value-based manufacturing optimisation in serverless clouds for industry 4.0," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '18. New York, NY, USA: ACM, 2018, pp. 1222–1229. [Online]. Available: http://doi.acm.org/10.1145/3205455.3205501

[34] M. W. Przewozniczek, R. Tinós, and M. M. Komarnicki, "First improvement hill climber with linkage learning – on introducing dark gray-box optimization into statistical linkage learning genetic algorithms," in *Proceedings of the 2023 Annual Conference on Genetic and Evolutionary Computation*, 2023 (in press).

[35] F. Chicano, G. Ochoa, D. Whitley, and R. Tinós, "Dynastic potential crossover operator," *Evolutionary Computation*, vol. online first, pp. 1–35, 2021.

[36] R. Tinós, D. Whitley, F. Chicano, and G. Ochoa, "Partition crossover for continuous optimization: epx," in *Proc. of GECCO'2021*, 2021.

[37] R. Tinós, M. W. Przewozniczek, and D. Whitley, "Iterated local search with perturbation based on variables interaction for pseudo-boolean optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 296–304. [Online]. Available: https://doi.org/10.1145/3512290.3528716

[38] K. Orphanou, D. Thierens, and P. A. N. Bosman, "Learning bayesian network structures with gomea," in *GECCO*, 2018.

[39] M. W. Przewozniczek and M. M. Komarnicki, "Empirical problem decomposition — the key to the evolutionary effectiveness in solving a large-scale non-binary discrete real-world problem," *Applied Soft Computing*, vol. 113, p. 107864, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494621007869

[40] S. Wozniak, M. W. Przewozniczek, and M. M. Komarnicki, "Parameterless population pyramid for permutation-based problems," in *Parallel Problem Solving from Nature – PPSN XVI*, T. Bäck, M. Preuss, A. Deutz, H. Wang, C. Doerr, M. Emmerich, and H. Trautmann, Eds. Cham: Springer International Publishing, 2020, pp. 418–430.

[41] M. W. Przewozniczek, M. M. Komarnicki, P. A. N. Bosman, D. Thierens, B. Frej, and N. H. Luong, "Hybrid linkage learning for permutation optimization with gene-pool optimal mixing evolutionary algorithms," in *GECCO*, 2021.

[42] M. M. Komarnicki, M. W. Przewozniczek, H. Kwasnicka, and K. Walkowiak, "Incremental recursive ranking grouping for large scale global optimization," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2022.

[43] N. H. Luong, H. L. Poutré, and P. A. Bosman, "Multi-objective gene-pool optimal mixing evolutionary algorithm with the interleaved multi-start scheme," *Swarm and Evolutionary Computation*, 2018.

[44] S.-H. Hsu and T.-L. Yu, "Optimization by pairwise linkage detection, incremental linkage set, and restricted / back mixing: DSMGA-II," in *GECCO*, 2015.

[45] C. A. Floudas and X. Lin, "Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications," *Annals of Operations Research*, vol. 139, no. 1, pp. 131–162, 2005.

[46] F. Rossi, P. Van Beek, and T. Walsh, "Constraint programming," *Foundations of Artificial Intelligence*, vol. 3, pp. 181–211, 2008.

[47] J. Clausen, "Branch and bound algorithms-principles and examples," *Department of computer science, University of Copenhagen*, pp. 1–30, 1999.

[48] K. Holmström, A. O. Göran, and M. M. Edvall, "User's guide for tomlab/cplex v12. 1," *Tomlab Optim. Retrieved*, vol. 1, p. 2017, 2009.

[49] L. Perron and V. Furnon, "Or-tools," Google. [Online]. Available: https://developers.google.com/optimization/

[50] V. Maniezzo, T. Stützle, and S. Voss, *Matheuristics – Hybridizing Metaheuristics and Mathematical Programming*, 01 2010.

[51] M. A. Boschetti and V. Maniezzo, "Matheuristics: using mathematics for heuristic design," *4or*, vol. 20, no. 2, pp. 173–208, 2022.

[52] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE signal processing magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[53] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[54] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of reinforcement learning and control*, pp. 321–384, 2021.

[55] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, ser. HotNets '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 50–56. [Online]. Available: https://doi.org/10.1145/3005745.3005750

[56] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6351–6361.

[57] R. Liu, R. Piplani, and C. Toro, "A deep multi-agent reinforcement learning approach to solve dynamic job shop scheduling problem," *Computers & Operations Research*, vol. 159, p. 106294, 2023.

[58] R. K. Phanden, A. Jain, and R. Verma, "Integration of process planning and scheduling: a state-of-the-art review," *International Journal of Computer Integrated Manufacturing*, vol. 24, no. 6, pp. 517–534, 2011. [Online]. Available: https://doi.org/10.1080/0951192X.2011.562543

[59] P. J. Van Laarhoven, E. H. Aarts, and J. K. Lenstra, "Job shop scheduling by simulated annealing," *Operations research*, vol. 40, no. 1, pp. 113–125, 1992.

[60] C. Pautasso and E. Wilde, "Restful web services: principles, patterns, emerging technologies," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 1359–1360.

[61] B. W. Goldman and W. F. Punch, "Parameterless population pyramid," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014.

[62] M. Li, S. Yang, and X. Liu, "Diversity comparison of pareto front approximations in many-objective optimization," *IEEE Trans. on Cybernetics*, vol. 44, no. 12, pp. 2568–2584, Dec 2014.

[63] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, "Modified distance calculation in generational distance and inverted generational distance," in *Evolutionary Multi-Criterion Optimization*, A. Gaspar-Cunha, C. Henggeler Antunes, and C. C. Coello, Eds. Cham: Springer International Publishing, 2015, pp. 110–125.

[64] E. Zitzler, D. Brockhoff, and L. Thiele, "The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2007, pp. 862–876.

[65] N. Gunantara, "A review of multi-objective optimization: Methods and its applications," *Cogent Engineering*, vol. 5, no. 1, p. 1502242, 2018.

[66] H. Ishibuchi, H. Masuda, and Y. Nojima, "A study on performance evaluation ability of a modified inverted generational distance indicator," in *Proceedings of the 2015 annual conference on genetic and evolutionary computation*, 2015, pp. 695–702.

[67] M. W. Przewozniczek, M. M. Komarnicki, and B. Frej, "Direct linkage discovery with empirical linkage learning," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021, pp. 609–617.

[68] K. Hogrefe, W. Ziegler, and G. Goldenberg, "Measuring the formal diversity of hand gestures by their hamming distance," *Integrating Gestures*, pp. 75–88, 2011.