An efficient discrete invasive weed optimization for blocking flow-shop scheduling problem

用于阻塞流水车间调度问题的有效离散入侵杂草优化

Zhongshi Shao [a] , Dechang Pi [a,b,*] , Weishi Shao [a] , Peisen Yuan [c]

汪中世 [a] ， 皮德昌 [a,b,*] ， 邵伟 [a] ， 袁培森 [c]

a College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

南京航空航天大学计算机科学与技术学院，南京，中国

[b] Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China

[b] 南京软件新技术与产业化协同创新中心，南京，中国

[c] College of Information Science and Technology, Nanjing Agricultural University, Nanjing, China

南京农业大学信息科学与技术学院，南京，中国

A R T I C L E I N F O

文章信息

Keywords:

关键词:

Flow-shop scheduling with blocking

带阻塞的流水车间调度

Makespan

最长完成时间

Invasive weed optimization

入侵杂草优化

Spatial dispersal

空间扩散

Local search

局部搜索

A B S T R A C T

摘要

This paper proposes a discrete invasive weed optimization (DIWO) to solve the blocking flow-shop scheduling problem (BFSP) with makespan criterion, which has important practical applications in modern industry. In the proposed DIWO, an effective heuristic and the random method are combined to generate an initial plant population with high quality and diversity. To keep the searching ability and efficiency, a random-insertion-based spatial dispersal is presented by means of the normal distribution. Moreover, a shuffle-based referenced local search is embedded to further enhance local exploitation ability. An improved competitive exclusion is developed to determine an offspring plant population with good quality and diversity. The parameters setting is investigated based on a design-of-experiment approach. The effectiveness and applicability of the proposed spatial dispersal and local search are confirmed through numerical comparisons. Finally, a comprehensive computational evaluation including several state-of-the-art algorithms, together with statistical analyses, show that the proposed DIWO algorithm produces better results than all compared algorithms by significant margin.

本文提出了一种离散的入侵杂草优化算法 (DIWO)，用于解决以最大完工时间 (makespan) 为标准的阻塞流水车间调度问题 (BFSP)，这在现代工业中具有重要的实际应用。在所提出的 DIWO 中，有效的启发式方法和随机方法相结合，生成高质量和多样性的初始种群。为了保持搜索能力和效率，通过正态分布提出了一种基于随机插入的空间扩散方法。此外，嵌入了一种基于洗牌的参考局部搜索，以进一步增强局部开发能力。开发了一种改进的竞争排斥方法，以确定具有良好质量和多样性的后代种群。参数设置是基于实验设计方法进行的。通过数值比较，验证了所提出空间扩散和局部搜索的有效性和适用性。最后，通过包括几种最先进算法的综合计算评估以及统计分析，表明所提出的 DIWO 算法比所有比较的算法产生了更好的结果，并有显著的边际优势。

1. Introduction

1. 引言

The blocking flow-shop scheduling problem (BFSP) (Pinedo, 2012) is an important branch of the traditional flow-shop scheduling problems. In such problem, due to technological requirements or process characteristics in some stages of manufacturing process, there are no intermediate buffers or buffers are not allowed between the adjacent machines so that the processed jobs must be kept in the current machines until the downstream machines are free (Wang et al., 2010b). Many industrial productive systems can be modeled as BFSP, e.g., chemical industry (Merchan and Maravelias, 2016; Ronconi, 2004), serial manufacturing processes (Koren et al., 2017), the production of concrete blocks (Grabowski and Pempera, 2000), robotic cell (Ribas et al., 2015), the iron and steel industry (Gong et al., 2010), etc. In view of the practical significance of the BFSP, it is essential to develop effective methods to solve it.

阻塞流水车间调度问题 (BFSP)(Pinedo，2012) 是传统流水车间调度问题的一个重要分支。在这样的问题中，由于制造过程中某些阶段的技术要求或工艺特性，相邻机器之间不存在中间缓冲区或者不允许设置缓冲区，因此处理过的任务必须保持在当前机器上，直到下游机器空闲 (Wang et al.，2010b)。许多工业生产系统可以被建模为 BFSP，例如，化工行业 (Merchan 和 Maravelias，2016；Ronconi，2004)、串联制造过程 (Koren et al.，2017)、混凝土砖的生产 (Grabowski 和 Pempera，2000)、机器人单元 (Ribas et al.，2015)、钢铁工业 (Gong et al.，2010) 等。鉴于 BFSP 的实践意义，开发有效的方法来解决它是至关重要的。

The BFSP is a typical NP-hard problem (Hall and Sriskandarajah, 1996) when the number of machines is larger than two. With the increase of the problem size, the BFSP becomes more and more complicated so that it is difficult to completely solve it. To tackle this challenge, much more effort has been dedicated to providing effective methods to find high-quality solutions instead of the optimal solutions in a reasonable computational time. These effective methods can be classified into two categories: constructive heuristics and improvement heuristics. For the constructive heuristics, some specific rules are used to assign a priority index to each job to construct a scheduling permutation. Some constructive heuristics have been proposed to solve the BFSP, e.g., Profile Fitting (PF) (Mccormick et al., 1989), MinMax (MM) (Ronconi, 2004), etc. To further get the better solutions, Ronconi (2004) incorporated the insertion procedure of NEH into the MM and PF heuristics to propose two improved heuristics, i.e., MME and PFE. Meanwhile, Ronconi showed that the performance of both MME and PFE was better than the original NEH over the problems with 500 jobs and 20 machines. Pan and Wang (2012) found that the idle time and blocking time on the earlier stage machines may lead to more delay for the successive jobs. According to this rule, two constructive heuristics with weighted values, i.e., wPF and PW were proposed. Meanwhile, in order to retain the good characteristics of the PF and PW heuristics, they also combined them with the partial NEH implementation to develop three new constructive heuristics including PF-NEH, wPF-NEH and PW-NEH. Besides those above, Wang et al. (2012) considered the average value and standard deviation of processing time to propose a modified NEH heuristic. Ribas et al. (2011) presented a modified NEH heuristic which considered the reversibility of the problems. Fernandez-Viagas et al. (2016) conducted a comprehensive evaluation of the available constructive heuristics for BFSP, and proposed a constructive heuristic based on beam search approach.

BFSP 是一个典型的 NP 难问题 (Hall 和 Sriskandarajah，1996)，当机器数量超过两台时。随着问题规模的增加，BFSP 变得越来越复杂，以至于难以完全解决。为了应对这一挑战，大量的努力已经投入到提供有效方法中，以在合理的计算时间内找到高质量解而非最优解。这些有效方法可以分为两类: 构造启发式和改进启发式。对于构造启发式，使用一些特定规则为每个作业分配优先级指数，以构建调度排列。已经提出了一些构造启发式来解决 BFSP，例如 Profile Fitting(PF)(Mccormick 等人，1989)、MinMax(MM)(Ronconi, 2004) 等。为了进一步获得更好的解，Ronconi(2004) 将 NEH 的插入过程整合到 MM 和 PF 启发式中，提出了两种改进的启发式方法，即 MME 和 PFE。同时，Ronconi 展示了在 500 个作业和 20 台机器的问题上，MME 和 PFE 的性能都优于原始的 NEH。Pan 和 Wang(2012) 发现，早期阶段机器上的空闲时间和阻塞时间可能会导致后续作业的更多延迟。根据这一规则，提出了两种带有加权值的构造启发式，即 wPF 和 PW。同时，为了保留 PF 和 PW 启发式的良好特性，它们还与部分 NEH 实现相结合，开发出三种新的构造启发式，包括 PF-NEH、wPF-NEH 和 PW-NEH。除了上述内容之外，Wang 等人 (2012) 考虑了处理时间的平均值和标准差，提出了一个修改后的 NEH 启发式。Ribas 等人 (2011) 提出了一种考虑问题可逆性的修改后 NEH 启发式。Fernandez-Viagas 等人 (2016) 对 BFSP 的可用构造启发式进行了全面评估，并提出了基于束搜索方法的构造启发式。

---

* Corresponding author.

* 通讯作者。

E-mail addresses: shaozhongshi@hotmail.com (Z. Shao), dc.pi@nuaa.edu.cn (D. Pi), shaoweishi@hotmail.com (W. Shao), peiseny@njau.edu.cn (P. Yuan).

电子邮件地址:shaozhongshi@hotmail.com (Z. Shao), dc.pi@nuaa.edu.cn (D. Pi), shaoweishi@hotmail.com (W. Shao), peiseny@njau.edu.cn (P. Yuan)。

For the improvement heuristics, many novel metaheuristics recently have been proposed and provided excellent results in acceptable computational time. Wang et al. (2010a) proposed a hybrid discrete differential evolution (HDDE) algorithm. In the HDDE, the new mutation and crossover operators were developed. A local search based on the insert neighborhood was embedded to enhance the local exploitation ability. A speedup method was presented to evaluate the insert neighboring solutions. This speedup method effectively improved the efficiency of the whole algorithm. Wang and Tang (2012) proposed a discrete particle swarm optimization (DPSO) in which a self diversity control strategy was adopted to diversify the population, and a stochastic variable neighborhood search was used to improve the search intensification. Lin and Ying (2013) proposed a revised artificial immune system (RAIS) based on the characteristics of artificial immune systems and the annealing process of simulated annealing algorithm. Pan et al. (2013b) proposed a high-performing memetic algorithm (MA) which was consisted of an effective heuristic combined PF with NEH, a path-relinking-based crossover operator, a referenced local search and a procedure of controlling diversity. Ribas et al. (2011, 2013) proposed two effective non-population-based algorithms: iterated greedy algorithm (IG) and competitive variable neighborhood search (SVNS) algorithm. Ding et al. (2016) investigated some new blocking properties of the BFSP and proposed an iterated greedy (IG) algorithm based on these priorities. Han et al. (2016a) proposed a modified fruit fly optimization (MFFO) algorithm in which a problem-specific heuristic, a smell-based search, and a speed-up insert-neighborhood-based local search were employed. Apart from the metaheuristics above, some methods were also presented to solve the BFSP with makespan criterion, e.g., an estimation of distribution algorithm (Jarboui et al., 2009), an improved artificial bee colony (IABC) (Han et al., 2011), a hybrid modified global-best harmony search (hmgHS) (Wang et al., 2011), a dynamic multi-swarm particle swarm optimizer (DMS-PSO) (Liang et al., 2011), a cluster enhanced differential evolution (EDEc) (Davendra et al., 2012), a three-phase algorithm (TPA) (Wang et al., 2012), a discrete self-organizing migrating algorithm (DSOMA) (Davendra and BialicDavendra, 2013), a discrete artificial bee colony algorithm incorporating differential evolution (DE-ABC) (Han et al., 2015b), a populated local search with differential evolution algorithm (DE_PLS) (Tasgetiren et al., 2015), a hybrid combinatorial particle swarm optimization algorithm (HCPSO) (Eddaly et al., 2016), iterated greedy algorithms (Tasgetiren et al., 2017), an estimation of distribution with path relinking (P-EDA) (Shao et al., 2018a), etc. Besides, many other effective metaheuristics have been also presented to solve other types of BFSP in recent years. Riahi et al. (2017) employed the scatter search to solve the mixed BFSP. Shao et al. (2018b) proposed an discrete water wave optimization (DWWO) algorithm to solve the BFSP with sequence-dependent setup times. Nouri and Ladhari (2017) proposed a genetic algorithm (MBGA) for BFSP with makespan and total flowtime. For the same problem, Deng et al. (2016) proposed a multi-objective discrete group search optimizer (MDGSO). Shao et al. (2017) presented a self-adaptive discrete invasive weed optimization (SaDIWO) to solve the BFSP with total tardiness criterion. Aiming at the same problem, Nagano et al. (2017) also presented an evolutionary clustering search (ECS) algorithm. Although many methods have been proposed and applied to solve variety of BFSPs, the pursuit of efficient methods should not be stopped.

对于改进启发式方法，最近提出了许多新颖的元启发式算法，并在可接受计算时间内提供了优秀的结果。王等人 (2010a) 提出了一种混合离散差分进化 (HDDE) 算法。在 HDDE 中，开发了新的变异和交叉操作符。基于插入邻域的局部搜索被嵌入，以增强局部开发能力。提出了一种加速方法来评估插入邻域解。这种加速方法有效地提高了整个算法的效率。王和唐 (2012) 提出了一种离散粒子群优化 (DPSO) 算法，在该算法中采用了自我多样性控制策略来多样化种群，并使用随机变量邻域搜索来提高搜索强化。林和应 (2013) 提出了一种基于人工免疫系统特征和模拟退火算法退火过程的改进人工免疫系统 (RAIS)。潘等人 (2013b) 提出了一种高性能的遗传算法 (MA)，该算法由有效的启发式方法结合 PF 和 NEH、基于路径连接的交叉操作符、参考局部搜索和多样性控制程序组成。里巴斯等人 (2011，2013) 提出了两种有效的非种群基础算法: 迭代贪婪算法 (IG) 和竞争性变量邻域搜索 (SVNS) 算法。丁等人 (2016) 研究了一些 BFSP 的新阻塞特性，并基于这些优先级提出了一个迭代贪婪 (IG) 算法。韩等人 (2016a) 提出了一种改进的果蝇优化 (MFFO) 算法，在该算法中使用了特定问题的启发式方法、基于气味的搜索和加速插入邻域的局部搜索。除了上述元启发式算法外，还提出了一些解决 BFSP 带有完工时间标准的方法，例如分布估计算法 (Jarboui 等人，2009)、改进的人工蜜蜂群算法 (IABC)(Han 等人，2011)、混合改进全局最佳和声搜索 (hmgHS)(Wang 等人，2011)、动态多群粒子群优化器 (DMS-PSO)(Liang 等人，2011)、聚类增强差分进化 (EDEc)(Davendra 等人，2012)、三阶段算法 (TPA)(Wang 等人，2012)、离散自组织迁移算法 (DSOMA)(Davendra 和 BialicDavendra，2013)、结合差分进化的离散人工蜜蜂群算法 (DE-ABC)(Han 等人，2015b)、带有差分进化算法的种群局部搜索 (DE_PLS)(Tasgetiren 等人，2015)、混合组合粒子群优化算法 (HCPSO)(Eddaly 等人，2016)、迭代贪婪算法 (Tasgetiren 等人，2017)、带有路径连接的分布估计 (P-EDA)(Shao 等人，2018a) 等。此外，近年来还提出了许多其他有效的元启发式算法来解决 BFSP 的其他类型。里亚希等人 (2017) 使用散射搜索解决混合 BFSP。Shao 等人 (2018b) 提出了一种解决带有序列相关设置时间的 BFSP 的离散水波优化 (DWWO) 算法。Nouri 和 Ladhari(2017) 为带有完工时间和总流程时间的 BFSP 提出了一种遗传算法 (MBGA)。对于同一问题，邓等人 (2016) 提出了一种多目标离散群搜索优化器 (MDGSO)。Shao 等人 (2017) 提出了一种解决带有总延迟标准的 BFSP 的自适应离散入侵杂草优化

(SaDIWO) 算法。针对同一问题，长野等人 (2017) 也提出了一种进化聚类搜索 (ECS) 算法。尽管已经提出了许多方法并应用于解决各种 BFSP，但追求高效方法的步伐不应停止。

Invasive weed optimization (IWO) is a swarm intelligence optimization algorithm, which was proposed by Mehrabian and Lucas (2006) for solving the solution of complex real world problems. IWO mimics the ecological behavior of colonizing weeds and has good exploration and exploitation ability in the search area. It shares many characteristics with evolutionary algorithms, but does not utilize evolution operators such as crossover and mutation. Instead, some interesting operators based on the features of weed plants, including reproduction, spatial dispersal, and competitive exclusion are employed. As a more robust, stochastic and derivative-free optimization tool (Barisal and Prusty, 2015), IWO has been successfully applied to solve many optimization problems, e.g., production scheduling (Sang et al., 2018a), non-uniform circular antenna arrays (Roy et al., 2011), economic load dispatch (Barisal and Prusty, 2015), traveling salesman problem (Zhou et al., 2015), unit commitment problem solution (Saravanan et al., 2014), portfolio optimization problem (Rezaei Pouya et al., 2016), distributed generation (Rama Prabha and Jayabarathi, 2016), antenna design (Das-tranj, 2017), vendor selection (Niknamfar and Niaki, 2018), optimal chiller loading (Zheng and Li, 2018), etc. Therefore, in view of the merits and extensive applications of IWO, we propose an effective discrete invasive weed optimization (DIWO) to solve the BFSP with makespan criterion in this paper. In the proposed algorithm, an effective heuristic and the random method are combined to generate an initial plant population. A random-insertion-based spatial dispersal is designed to produce the new individuals with high quality. A shuffle-based referenced local search is incorporated to reinforce the exploitation ability. Finally, to obtain an offspring plant population with high quality and diversity, an improved competitive exclusion is employed. Additionally, to the best of our knowledge, this work is the first reported application of IWO to solve the BFSP with makespan criterion. The major contribution and innovation of our work can be reflected in algorithm design and experimental results.

| 入侵杂草优化 (IWO) 是一种群体智能优化算法，由 Mehrabian 和 Lucas(2006) 提出，用于解决复杂现实世界问题的解决方案。IWO 模拟了入侵杂草的生态行为，在搜索区域内具有良好的探索和开发能力。它与进化算法有许多共同特征，但并不使用诸如交叉和突变之类的进化操作符。相反，它采用了一些基于杂草植物特征的有趣操作符，包括繁殖、空间分散和竞争排斥。作为一种更健壮、随机且无需导数的优化工具 (Barisal 和 Prusty，2015)，IWO 已成功应用于解决许多优化问题，例如生产调度 (Sang 等人，2018a)、非均匀圆形天线阵列 (Roy 等人，2011)、经济负荷调度 (Barisal 和 Prusty，2015)、旅行商问题 (Zhou 等人，2015)、单元承诺问题解决方案 (Saravanan 等人，2014)、投资组合优化问题 (Rezaei Pouya 等人，2016)、分布式发电 (Rama Prabha 和 Jayabarathi，2016)、天线设计 (Das-tranj，2017)、供应商选择 (Niknamfar 和 Niaki，2018)、最优冷水机组加载 (Zheng 和 Li，2018) 等。因此，鉴于 IWO 的优点和广泛的应用，本文提出了一种有效的离散入侵杂草优化 (DIWO) 算法，用于解决以最短完工时间标准为准则的 BFSP 问题。在所提出的算法中，将有效的启发式方法和随机方法相结合，生成初始植物种群。设计了一种基于随机插入的空间分散方法，以产生高质量的新个体。融入了一种基于洗牌的参考局部搜索方法，以增强开发能力。最终，为了获得高质量和多样性的后代植物种群，采用了改进的竞争排斥。此外，据我们所知，这项工作是首次报告使用 IWO 解决以最短完工时间标准为准则的 BFSP 问题的应用。我们工作的主要贡献和创新可以体现在算法设计和实验结果中。

- A discrete invasive weed optimization (DIWO) is proposed for solving BFSP with makespan criterion in this study. The phases of reproduction, spatial dispersal, and competitive exclusion are newly customized. Especially, a random-insertion-based spatial dispersal is designed to make the original IWO for solving the continuous optimization problems apply to the discrete optimization problem. More importantly, we incorporate a shuffle-based referenced local search into DIWO, which effectively enhances the performance of the algorithm. This new phase can be regarded as a supplement for the framework of the original IWO, which also provides an idea for further improving the performance of IWO.

- 在本研究中，提出了一种离散入侵杂草优化 (DIWO) 算法，用于解决以最短完工时间标准为目标的 BFSP 问题。对繁殖、空间扩散和竞争排斥等阶段进行了新的定制。特别是，设计了一种基于随机插入的空间扩散方法，使得原本用于解决连续优化问题的 IWO 算法能够适用于离散优化问题。更重要的是，我们将基于洗牌的参考局部搜索融入 DIWO 中，有效提升了算法的性能。这个新阶段可以看作是对原 IWO 框架的补充，也为进一步改进 IWO 的性能提供了思路。

- IWO is a nature-inspired optimization algorithm in artificial intelligence, which is initially used to solve the continuous optimization problems on engineering area. In this study, the IWO is successfully extended to solve a typical combinatorial optimization problem, which causes an enhancement in understanding of the searching behavior of IWO, and further enriches its applications. Meanwhile, the proposed discrete IWO incorporating distinct reproduction, spatial dispersal and competitive exclusion mechanism can also provide a solution for solving the similar engineering problems.

- IWO 是一种受自然启发的人工智能优化算法，最初用于解决工程领域的连续优化问题。在本研究中，IWO 成功地被扩展用于解决一个典型的组合优化问题，这增强了对 IWO 搜索行为的理解，并

进一步丰富了其应用。同时，所提出的离散 IWO 融合了独特的繁殖、空间扩散和竞争排斥机制，也能为解决类似的工程问题提供解决方案。

- The performance of the proposed DIWO algorithm was evaluated and compared with several high-performing methods in recent literature. The experimental results demonstrated that our proposed algorithm was superior to these methods. Additionally, we have summarized the best solutions of 14 recent state-of-the-art algorithms and compared with the proposed DIWO. The comparison results show that the proposed DIWO updates 47 best solutions out of 120 Taillard’ s benchmark instances (Taillard, 1993). The job processing sequences of these new best solutions are provided on the online supplementary material, which can be a reference for future work along this line of research.

- 所提出的 DIWO 算法的性能通过与近期文献中的几种高性能方法进行了评估和比较。实验结果表明，我们提出的算法优于这些方法。此外，我们总结了 14 种近期最先进的算法的最佳解决方案，并与提出的 DIWO 进行了比较。比较结果显示，提出的 DIWO 在 120 个 Taillard 基准实例 (Taillard, 1993) 中更新了 47 个最佳解决方案。这些新最佳解决方案的作业处理序列已提供在线补充材料，这可以作为未来沿此研究方向工作的参考。

The remainder of this paper is organized as follows. In Section 2, the BFSP problem is described. Next, the basic IWO is presented in Section 3. Section 4 elaborates on the proposed DIWO algorithm. Section 5 gives a full performance evaluation and comparison. Finally, Section 6 provides the concluding remarks and future research directions.

本文其余部分的组织结构如下。第 2 节描述了 BFSP 问题。接下来，第 3 节介绍了基本的 IWO 算法。第 4 节详细阐述了提出的 DIWO 算法。第 5 节进行了全面的性能评估和比较。最后，第 6 节提供了结论性意见和未来研究方向。

2. Blocking flow-shop scheduling problem

2. 阻塞流水车间调度问题

The blocking flow-shop scheduling problem (BFSP), denoted as Fm|blocking| $C_{\max}$ according to Graham et al. (1979) can be briefly described as: n jobs have to be processed m machines with the same permutation on each machine. There are no intermediate buffers between any two consecutive machines. Due to no buffers between the adjacent machines, a job having its operation on the current machine cannot leave the machine until the next machine is free for processing. That is to say, if the downstream machine is busy, the job must be blocked in the current machine. Additionally, other constraints should be considered for Fm | blocking | $C_{\max}$, which are described as follows:

阻塞流水车间调度问题 (BFSP)，根据 Graham 等人 (1979 年) 表示为 Fm|blocking| $C_{\max}$ ，可以简要描述为: n 作业必须在 m 台机器上按照相同的排列顺序进行处理。任何两个连续机器之间没有中间缓冲区。由于相邻机器之间没有缓冲区，因此在当前机器上进行的作业不能离开该机器，直到下一台机器空闲可以进行加工。也就是说，如果下游机器忙碌，作业必须在当前机器上被阻塞。此外，还应考虑 Fm | blocking | $C_{\max}$ 的其他约束，具体描述如下:

- Each job can be only processed by one machine at a time.
- 每个作业一次只能由一台机器处理。
- Each machine can process only one job at a time.
- 每台机器一次只能处理一个作业。
- The processing time of each job on each machine is predetermined.
- 每个作业在每台机器上的加工时间是预定的。
- The setup time and transportation time are assumed to be included in the processing time.
- 假设设置时间和运输时间包含在加工时间内。
- Preemption is not allowed.
- 不允许抢占。

The purpose considered in Fm | blocking | $C_{\max}$ is to obtain a permutation for processing all jobs on all machines so that its maximum completion time (makespan, denoted as $C_{\max}$) is minimized. Let $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ be a permutation sequence. $p_{\pi(i),j}$ is the processing time of job $\pi(i)$ on machine $j.d_{\pi(i),j}$ denotes the departure time of job $\pi(i)$ on machine j . According to Ronconi (2004), $d_{\pi(i),j}$ can be calculated as follows:

在 Fm | 阻止 | $C_{\max}$ 中考虑的目标是获取一个处理所有机器上所有工作的排列，使得其最大完成时间 (称为 $C_{\max}$) 最小化。设 $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ 为一个排列序列。$p_{\pi(i),j}$ 是工作 $\pi(i)$ 在机器 $j.d_{\pi(i),j}$ 上的处理时间，j 表示工作 $\pi(i)$ 在机器 $d_{\pi(i),j}$ 上的出发时间。根据 Ronconi (2004) 的研究，$d_{\pi(i),j}$ 可以按以下方式计算：

$$d_{\pi(1),0} = 0 \tag{1}$$

$$d_{\pi(1),j} = d_{\pi(1),j-1} + p_{\pi(1),j} \; j = 1, \ldots, m-1 \tag{2}$$

$$d_{\pi(i),0} = d_{\pi(i-1),1} \; i = 2, \ldots, n \tag{3}$$

$$d_{\pi(i),j} = \max\left\{d_{\pi(i),j-1} + p_{\pi(i),j}, d_{\pi(i-1),j+1}\right\} \; i = 2, \ldots, n$$

$$j = 1, \ldots, m-1 \tag{4}$$

$$d_{\pi(i),m} = d_{\pi(i),m-1} + p_{\pi(i),m} \; i = 1, \ldots, n \tag{5}$$

where $d_{\pi(1),0}$ denotes the beginning time of processing. Then, the makespan or the maximum completion time of the schedule $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ is $C_{\max}(\pi) = \max\limits_{i=1,2,\ldots,n} d_{\pi(i),m} = d_{\pi(n),m}$. Therefore, the BFSP with makespan criterion is to find a job permutation $\pi^* \in \Pi$ such that $C_{\max}(\pi^*) = \min\limits_{\pi \in \Pi} C_{\max}(\pi)$, where Π is the set of all job permutations.

其中 $d_{\pi(1),0}$ 表示处理的开始时间。那么，调度 $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ 的最大完成时间 (即工期) 是 $C_{\max}(\pi) = \max\limits_{i=1,2,\ldots,n} d_{\pi(i),m} = d_{\pi(n),m}$ 。因此，以最大完工时间 (工期) 为标准的最佳排列搜索 (BFSP) 是要找到一个工作排列 $\pi^* \in \Pi$ ，使得 $C_{\max}(\pi^*) = \min\limits_{\pi \in \Pi} C_{\max}(\pi)$ ，其中 Π 是所有工作排列的集合。

3. Introduction to the basic IWO algorithm

3. 基本 IWO 算法介绍

The Invasive Weed Optimization (IWO) is a swarm intelligence metaheuristic, which mimics the natural behavior of weeds in colonizing and finding suitable place for growth and reproduction (Basak et al., 2013). Some interesting properties of weeds that are invasive, fast reproduction, distribution and competitive exclusion are taken advantages by the IWO. The basic IWO is composed of four basic steps: initialization, reproduction, spatial dispersal and competitive exclusion. They are described as follows (Mehrabian and Lucas, 2006):

侵略性杂草优化 (IWO) 是一种群体智能启发式算法，它模仿了杂草在殖民和寻找适合生长和繁殖地点的自然行为 (Basak et al., 2013)。IWO 利用了一些杂草的有趣特性，如侵略性、快速繁殖、分布和竞争性排斥。基本 IWO 由以下四个基本步骤组成：初始化、繁殖、空间扩散和竞争性排斥。以下是它们的描述 (Mehrabian 和 Lucas, 2006):

(1) Initialization. A population **POP** $= \{X_1, X_2, \ldots, X_{N_0}\}, X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,D}]$ that has N_0 individuals (weeds) is randomly generated over the D -dimensional search space.

(1) 初始化。在 **POP** $= \{X_1, X_2, \ldots, X_{N_0}\}, X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,D}]$ 的 N_0 个个体 (杂草) 随机生成于 D 维搜索空间中。

(2) Reproduction. Each weed of plant population can generate some seeds within a specified region centered at its own position. The number of seeds for each weed is determined by its relative fitness with respect to the best and the worst fitness, which can be calculated by:

(2) 繁殖。每个植物种群中的杂草在其自身位置中心的指定区域内可以产生一些种子。每株杂草的种子数量由其相对于最佳和最差适应度的相对适应度决定，计算公式如下:

$$S_i = \left\lfloor \frac{f_{\max} - f_i}{f_{\max} - f_{\min}} \times (S_{\max} - S_{\min}) + S_{\min} \right\rfloor \tag{6}$$

where S_i is the number of seeds for i th weed. f_i is the fitness value of i th weed. $f_{\min}$ and $f_{\max}$ denote the fitness values of the best and worst weeds among the current population, respectively. $S_{\min}$ and $S_{\max}$ denote the minimum and maximum number of seeds that allowed to be generated for each weed, respectively. It is clear from Eq. (6) that the higher the fitness f_i is, the more seeds it produces.

其中 S_i 是第 i 株杂草的种子数量。f_i 是第 i 株杂草的适应度值。$f_{\min}$ 和 $f_{\max}$ 分别表示当前种群中最佳和最差杂草的适应度值。$S_{\min}$ 和 $S_{\max}$ 分别表示允许为每株杂草产生的最小和最大种子数量。从等式 (6) 可以明显看出，适应度 f_i 越高，产生的种子越多。

(3) Spatial dispersal. The generated seeds are randomly scattered over the D -dimensional search space by means of a normal distribution with mean zero and varying standard variance σ . The normal distribution ensures that the generated seeds are scattered near to their parents. σ adaptively decreases from $\sigma_{\max}$ to $\sigma_{\min}$ with the increase of generation so that the algorithm can gradually move from exploration to exploitation. The standard variance for the k th iteration σ_k is given by:

(3) 空间扩散。产生的种子通过均值为零且标准差不同的正态分布随机散布在 D 维搜索空间中。正态分布确保产生的种子靠近其亲本。σ 随着代数的增加自适应地从 $\sigma_{\max}$ 减少到 $\sigma_{\min}$ ，以便算法能够从探索逐渐过渡到利用。第 k 次迭代的标准差 σ_k 由以下公式给出:

$$\sigma_k=\left(\frac{iter_{\max}-k}{iter_{\max}}\right)^n\times\left(\sigma_{\max}-\sigma_{\min}\right)+\sigma_{\min} \tag{7}$$

where n is a nonlinear modulation index. iter $r_{\max}$ is the maximum number of iterations. k is the current iteration number. In addition, according to the standard variance σ_k above, the location of a seed are determined by:

其中 n 是一个非线性调制指数。iter $r_{\max}$ 是最大迭代次数。k 是当前迭代次数。此外，根据上述标准差 σ_k ，种子的位置由以下公式确定:

$$x_{i,j}^{\prime}=x_{i,j}+N\left(0,\sigma_k^2\right) \tag{8}$$

where $x_{i,j}$ is the j th dimension of the i th weed in current population. $N\left(0,\sigma_k^2\right)$ is a sampling function that returns a normally distributed random number with mean zero and the standard deviation σ_k

其中 $x_{i,j}$ 是当前种群中第 i 株杂草的第 j 维。$N\left(0,\sigma_k^2\right)$ 是一个采样函数，返回均值为零且标准差为 σ_k 的正态分布随机数。

(4) Competitive exclusion. After all seeds have found their locations over the search area, the new seeds would grow to weeds. Initially, the weeds in a colony will reproduce fast and all produced weeds will be reserved in the colony. When the number of weeds in a colony reaches its maximum value $(P_{\max})$, all new created weeds and their parental weeds are ranked together according to their fitness. The weaker and inappropriate weeds are eliminated so that the colony can maintain the maximum allowable population. This mechanism gives a chance for lower fitness plants to reproduce. If their offspring individuals have good fitness, then they can survive. This procedure is named competitive exclusion and is the selection procedure of IWO.

(4) 竞争排斥。在所有种子在搜索区域内找到位置之后，新的种子将生长为杂草。最初，群落中的杂草会快速繁殖，并且所有产生的杂草都会被保留在群落中。当群落中的杂草数量达到最大值 $(P_{\max})$ 时，所有新产生的杂草及其亲本杂草将根据它们的适应度一起排名。较弱的和不适当的杂草将被淘汰，以便群落能够维持最大允许的种群数量。这个机制为适应度较低的植物提供了繁殖的机会。如果它们的后代个体具有好的适应度，那么它们就能够存活。这个过程被称为竞争排斥，是 IWO 的选择过程。

4. DIWO for BFSP

4. DIWO 用于 BFSP

From the procedure of the basic IWO above, it can be seen that IWO can be directly applied only when solving continuous optimization problems. That is to say, when solving the discrete optimization problems, a discrete variant of IWO must be developed. Therefore, a discrete IWO (DIWO) is presented to solve BFSP with makespan criterion in this section. The proposed DIWO algorithm contains four main components: a population scheme combining an effective heuristic and the random method to produce an initial plant population with high quality and diversity, a spatial dispersal model based on random insertion to maintain the searching ability, a competitive exclusion mechanism to determine the offspring individuals and an effective local search procedure to enhance the performance of DIWO. With these effective and advanced technologies, the DIWO is expected to generate high quality solutions with robustness for BFSP. In the following sections, each component of DIWO is firstly elaborated, and then we outline the complete procedure of DIWO.

从上述基本 IWO 的过程可以看出，只有解决连续优化问题时才能直接应用 IWO。也就是说，在解决离散优化问题时，必须开发 IWO 的离散变体。因此，在本节中，提出了一种离散 IWO(DIWO)，用于在工期准则下解决 BFSP。所提出的 DIWO 算法包含四个主要组成部分: 一种结合有效启发式和随机方法的种群方案，以产生高质量和多样性的初始植物种群，一种基于随机插入的空间扩散模型以保持搜索能力，一种确定后代个体的竞争排斥机制，以及一种有效的局部搜索过程以提高 DIWO 的性能。有了这些有效和先进的技术，DIWO 预计将为 BFSP 生成具有鲁棒性的高质量解决方案。在接下来的章节中，我们将首先详细阐述 DIWO 的每个组成部分，然后概述 DIWO 的完整过程。

4.1. Solution representation and initialization

4.1. 解的表示和初始化

The permutation-based-jobs representation is used to encode individuals, which has been frequently adopted in the literature (Han et al., 2016a; Ribas et al., 2015; Shao and Pi, 2016). Given a permutation of jobs, the jobs are successively scheduled from left to right onto machines. To generate an initial plant population with a certain level of quality and diversity, a well-known heuristic named PF-NEH(x)(Pan and Wang, 2012) is firstly employed to produce a weed with high quality. Then, the rest weeds are randomly generated in the entire colony. PF-NEH(x)not only considers the total idle and blocking time but also the different initial jobs. The procedure of initialization is described in Algorithm 1.

基于排列的工作表示法被用来编码个体，这在文献中经常被采用 (Han et al., 2016a; Ribas et al., 2015; Shao 和 Pi, 2016)。给定一个工作的排列，工作会依次从左到右调度到机器上。为了生成具有一定质量和多样性的初始种群，首先使用了一个著名的启发式算法 PF-NEH(x)(Pan 和 Wang, 2012) 来生成高质量的首个个体。然后，其余个体在整个种群中随机生成。PF-NEH(x) 不仅考虑了总的空闲和阻塞时间，还考虑了不同的初始工作。初始化过程在算法 1 中描述。

Following Pan and Wang (2012), the PF-NEH(x)with $\lambda = 25$ and $x = 5$ can obtain the best performance. Moreover, it should be noted that the speedup method proposed by Wang et al. (2010a) is used to evaluate the $((2n - \lambda + 1)\lambda)/2$ partial sequences in the insertion phase of the NEH heuristic, so the computational complexity of NEH can be reduced from $O(mn^3)$ to $O(mn^2)$. Since x sequences are generated in PF-NEH(x), the computational complexity is $O(xmn^2)$.

根据 Pan 和 Wang (2012)，带有 $\lambda = 25$ 和 $x = 5$ 的 PF-NEH(x) 能够获得最佳性能。此外，应注意 Wang et al. (2010a) 提出的加速方法被用于评估 NEH 启发式插入阶段的 $((2n - \lambda + 1)\lambda)/2$ 部分序列，因此 NEH 的计算复杂度可以从 $O(mn^3)$ 降低到 $O(mn^2)$ 。由于 PF-NEH(x) 生成了 x 序列，计算复杂度是 $O(xmn^2)$ 。

```
Algorithm 1. Population initialization
算法 1. 种群初始化
Input: $N_0, \lambda$ and $x$
输入: $N_0, \lambda$ 和 $x$
Output: POP
输出:POP
: **POP** $\leftarrow \varnothing$ ;
$\pi_I \leftarrow$ Generate an initial order of all jobs according to their non-decreasing total processing time;
$\pi_I \leftarrow$ 根据所有工作的非递增总处理时间生成一个初始顺序;
$k \leftarrow 1, □ \leftarrow \varnothing$ ;
repeat
重复
Set $\pi'(1) = \pi_I(k)$ and the unscheduled job set $\mathbf{U} = \mathbf{J} \setminus \{\pi_I(k)\}$ , where **J** is the job set;
设置 $\pi'(1) = \pi_I(k)$ 和未调度工作集 $\mathbf{U} = \mathbf{J} \setminus \{\pi_I(k)\}$ ，其中 **J** 是工作集;
Calculate the departure time $d_{1,j}$ for the position 1 of $\pi'$ , where $j = 1, \ldots, m$ ;
计算位置 1 的 $\pi'$ 的出发时间 $d_{1,j}$ ，其中 $j = 1, \ldots, m$ ;
$i \leftarrow 2$ ;
while $U \neq \varnothing$ do
当 $U \neq \varnothing$ 时
Compute the departure time $d_{i,j}$ of each job in **U** for position $i, j = 1, \ldots, m$ ;
计算位置 $i, j = 1, \ldots, m$ 的 **U** 中每个工作的出发时间 $d_{i,j}$ ;
Compute the sum of the idle time and blocking time of each job in **U** for position $i$ by:
计算每个工作在 **U** 中的空闲时间和阻塞时间的总和，位置 $i$ 通过以下方式:
```

```
$v_l = \sum_{j=1}^{m} (d_{i,j} - d_{i-1,j} - p_{\pi_l,j})\ l \in \mathbf{U}$ (9)
$\pi'(i) \leftarrow$ Select the job with the smallest of $v_l$ in $\mathbf{U}$ ;
$\pi'(i) \leftarrow$ 在 $\mathbf{U}$ 中选择具有最小 $v_l$ 的工作;
$i \leftarrow i+1$ and $\mathbf{U} \leftarrow \mathbf{U} \setminus \{\pi_{\min v_l}\}$ ;
$i \leftarrow i+1$ 和 $\mathbf{U} \leftarrow \mathbf{U} \setminus \{\pi_{\min v_l}\}$ ;
end while
结束循环
$\pi'' \leftarrow [\pi'(1), \pi'(2), \ldots, \pi'(n-\lambda)]$ ;
for $q = n - \lambda + 1$ to $n$ do
对于 $q = n - \lambda + 1$ 到 $n$ 进行循环
Take the job $\pi'(q)$ from $\pi'$ and test it into all possible positions of $\pi''$ ;
从 $\pi'$ 中取出工作 $\pi'(q)$ 并将其测试到 $\pi''$ 的所有可能位置;
Insert the job $\pi'(q)$ into the position of $\pi''$ with the lowest makespan;
将工作 $\pi'(q)$ 插入位置 $\pi''$ 中，使得最短完工时间 (makespan) 最低;
end for
结束循环
$k \leftarrow k+1$ and □ $\leftarrow$ □ $\cup \{\pi''\}$ ;
$k \leftarrow k+1$ 和 □ $\leftarrow$ □ $\cup \{\pi''\}$ ;
3: until $k == x$
3: 直到 $k == x$
$\pi_{\text{best}} \leftarrow$ Select the best one from □ ;
$\pi_{\text{best}} \leftarrow$ 从 □ 中选择最佳的一个;
POP $\leftarrow$ POP $\cup \{\pi_{\text{best}}\}$
POP $\leftarrow$ POP $\cup \{\pi_{\text{best}}\}$
repeat
重复
$\pi \leftarrow$ Randomly produce a weed in the entire colony;
$\pi \leftarrow$ 在整个群落中随机生成一个杂草;
If $\mathbf{POP} \cap \{\pi\} == \varnothing$ , then $\mathbf{POP} \leftarrow \mathbf{POP} \cup \{\pi\}$ ;
如果 $\mathbf{POP} \cap \{\pi\} == \varnothing$ ，那么 $\mathbf{POP} \leftarrow \mathbf{POP} \cup \{\pi\}$ ;
6: until $|\mathbf{POP}| == N_0$ .
6: 直到 $|\mathbf{POP}| == N_0$ 。
```

4.2. Reproduction

4.2. 繁殖

In the procedure of the basic IWO, the number of seeds generated by each weed is determined by its fitness. The higher the fitness is, the larger the number is. However, since the BFSP is a discrete optimization problem and the solution space is limited, many different permutations have the same fitness. When the fitness of each individual in population is same but they have different permutation sequences, the best fitness is equal to the worst fitness so that Eq. (6) is illegal. Actually, this illegal case often appears when solving the small instances, it is because the solution space is relatively small so that the individuals in the population quickly converge to a local or global optimum. Therefore, to circumvent such minor problem, the value of S_i produced by the weed π_i is determined by the following formula:

在基本 IWO 算法中，每株杂草生成的种子数量由其适应度决定。适应度越高，数量越大。然而，由于 BFSP 是一个离散优化问题，解空间有限，许多不同的排列具有相同的适应度。当种群中每个个体的适应度相同但它们的排列序列不同，最佳适应度等于最差适应度，因此式 (6) 是不合法的。实际上，在解决小规模实例时，这种不合法情况经常出现，这是因为解空间相对较小，因此种群中的个体迅速收敛到局部或全局最优。因此，为了避免此类小问题，杂草 π_i 产生的 S_i 值由以下公式确定:

$$S_i = \left\lfloor \frac{C_{\max}(\pi_{\text{worst}}) - C_{\max}(\pi_i) + \epsilon}{C_{\max}(\pi_{\text{worst}}) - C_{\max}(\pi_{\text{best}}) + \epsilon} \times (S_{\max} - S_{\min}) + S_{\min} \right\rfloor \tag{10}$$

where ϵ is the smallest constant in the computer, which is used to avoid division-by-zero. π_{worst} and π_{best} denote the worst and best weeds in the current plant population, respectively. $S_{\min}$ and $S_{\max}$ denote the minimum and maximum number of seeds, respectively. $f(\pi_i)$ denotes the makespan of π_i .

其中 ϵ 是计算机中最小的常数，用于避免除以零。π_{worst} 和 π_{best} 分别表示当前植物种群中最差和最好的杂草。$S_{\min}$ 和 $S_{\max}$ 分别表示最小和最大的种子数量。$f(\pi_i)$ 表示 π_i 的总完工时间。

4.3. Random-insertion-based spatial dispersal

4.3. 基于随机插入的空间分散

In the basic IWO, the dispersion of new seeds to the whole colony primarily depends on the normal distribution with mean zero and varying standard deviation. The varying standard deviation σ_k has a huge effect on the performance of the algorithm. However, such mechanism has its own shortcomings. The first one is, all weeds have same σ_k so that the spread scope of all new generated seed is same. As a result, the weeds with lower fitness may not get more opportunities to evolve to the better ones, while the weeds near the global optimum may be trapped into local optimum. The second one is, the maximum CPU execution time is usually adopted as the stopping rule in the many scheduling literature (Ding et al., 2016; Pan et al., 2013b; Ribas et al., 2015), whereas the standard deviation σ_k of the basic IWO algorithm is adjusted based on the iterations. Moreover, it is difficult to exactly predict the number of iterations in the finite execution time. Therefore, a mechanism based on fitness and execution time is proposed to overcome these shortcomings. Actually, the computational time of each iteration is approximate, so we can estimate an approximate total number of iterations. On the basis of this principle, a decreased σ_k with execution time is firstly given in Eq. (11) to solve the second problem.

在基本的 IWO 中，新种子的分散到整个群体主要依赖于均值为零且标准差不同的正态分布。变化的标准差 σ_k 对算法的性能有着巨大的影响。然而，这种机制有其自身的缺点。第一个缺点是，所有杂草具有相同的 σ_k，因此所有新生成种子的传播范围相同。结果，适应性较低的杂草可能没有更多的机会进化成更好的，而接近全局最优的杂草可能会陷入局部最优。第二个缺点是，在许多调度文献中 (Ding et al., 2016; Pan et al., 2013b; Ribas et al., 2015)，通常采用最大的 CPU 执行时间作为停止规则，而基本 IWO 算法的标准差 σ_k 是基于迭代次数进行调整的。此外，在有限的执行时间内准确预测迭代次数是困难的。因此，提出了一种基于适应性和执行时间的机制来克服这些缺点。实际上，每次迭代的时间是近似的，因此我们可以估计一个近似的总迭代次数。基于这一原则，首先在公式 (11) 中给出了随执行时间减少的 σ_k，以解决第二个问题。

$$\sigma_k = \left(1 - \frac{t - t_0}{t_{\max}}\right) \times (\sigma_{\max} - \sigma_{\min}) + \sigma_{\min} \tag{11}$$

where t is the current time. $t_{\max}$ is the maximum CPU execution time. t_0 is the beginning time of the algorithm. $\sigma_{\min}$ and $\sigma_{\max}$ denote the minimum and maximum values of σ_k, respectively. When any weeds are close to the potential global optimum, σ_k would decrease as Eq. (11) so that it does not miss the position of true global optimum. Meanwhile, the weeds with lower fitness should have larger spread scope to explore other promising areas. Thus, for the i th weed of the plant population in the k th iteration, $\sigma_{i,k}$ is calculated by:

其中 t 是当前时间。$t_{\max}$ 是最大的 CPU 执行时间。t_0 是算法开始时间。$\sigma_{\min}$ 和 $\sigma_{\max}$ 分别表示 σ_k 的最小值和最大值。当任何杂草接近潜在的全球最优解时，σ_k 会根据公式 (11) 减小，以便不会错过真实全局最优解的位置。同时，适应度较低的杂草应该有更大的扩散范围来探索其他有希望的领域。因此，在植物群体的第 i 次迭代中，第 k 株杂草的 $\sigma_{i,k}$ 是通过以下方式计算的:

$$\sigma_{i,k} = \begin{cases} \sigma_k & f(\pi_i) < f(\pi_{\text{median}}) \\ \sigma_k \times \left(\frac{C_{\max}(\pi_i) - C_{\max}(\pi_{\text{median}})}{C_{\max}(\pi_{\text{worst}}) - C_{\max}(\pi_{\text{median}}) + \varepsilon} \times 0.5 + 1\right) & f(\pi_i) > f(\pi_{\text{median}}) \end{cases}$$

(12)where π_{median} denotes the median element of the sorted plant population (sorted ascending according to fitness values). π_{worst} is the worst weed in the current plant population. ε is the smallest number in the computer, which is used to avoid division-by-zero. As can be easily observed from Eq. (12), when the makespan of π_i is smaller than $C_{\max}(\pi_{\text{median}})$, $\sigma_{i,k}$ is set to 1.5 times σ_k, which gives a chance to the worse weeds to spread their weeds over wider range and find more promising individuals.

其中 π_{median} 表示按适应度值升序排序的植物群体中的中位数元素。π_{worst} 是当前植物群体中最差的杂草。ε 是计算机中的最小数，用于避免除以零。从公式 (12) 可以容易地看出，当 π_i 的完工时间小于 $C_{\max}(\pi_{\text{median}})$, $\sigma_{i,k}$ 时，$C_{\max}(\pi_{\text{median}})$, $\sigma_{i,k}$ 被设置为 σ_k 的 1.5 倍，这给了较差的杂草一个机会，让它们的杂草在更广泛的范围内传播，并找到更多有希望的个体。

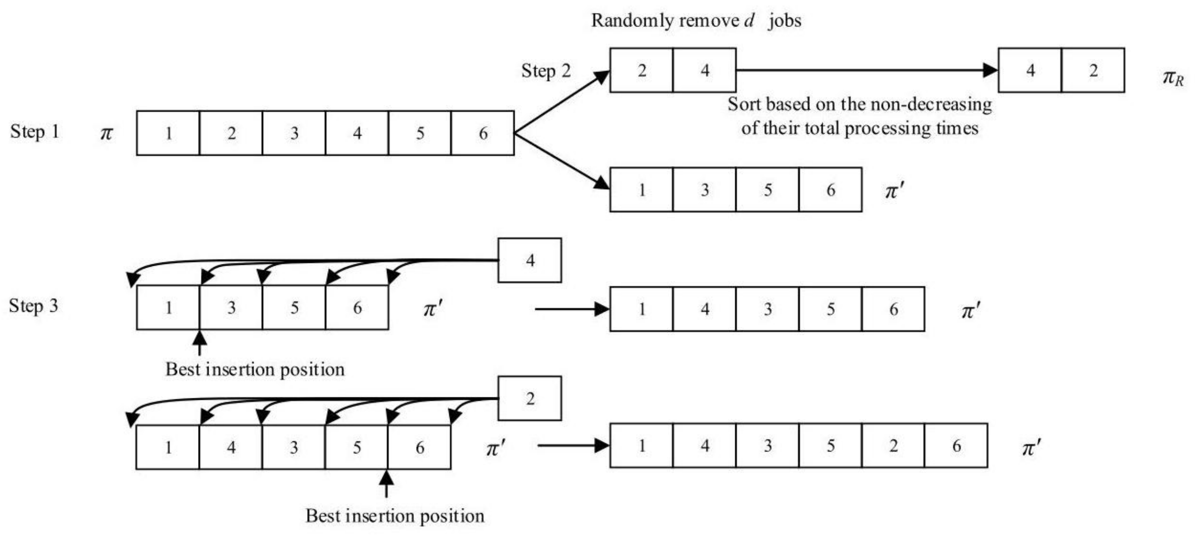

Fig. 1. Numerical example of the random insertion neighborhood.
图 1. 随机插入邻域的数值示例。

Since the BFSP is a discrete problem and the solutions are represented by the job permutations, the spatial dispersal of the basic IWO is not suitable. Therefore, we develop a discrete spatial dispersal. To be specific, in our proposed spatial dispersal, a random insertion neighborhood is designed to generate a new seed for each weed, which has three steps. Firstly, d jobs are randomly selected and removed from the solution (weed) π without repetition to form a partial sequence π_R . The remainder jobs in π form another partial sequence π' . Afterwards, according to Wang et al. (2011), the jobs with higher total processing times may cause to block their successive jobs and yield larger blocking times than less total processing times. The blocking time would increase the makespan value. The jobs with less total processing time should have high priorities. So we sort the jobs in π_R with the non-decreasing of their total processing time. Lastly, the jobs in π_R are successively re-inserted into π' by using the insertion of the NEH heuristic until a complete solution π' of n jobs is re-established. For the insertion position of each job, all possible insertion positions of π' are evaluated and the best one is chosen.

由于 BFSP 是一个离散问题，解决方案由作业排列表示，因此基本 IWO 的空间分散不适用。因此，我们开发了一种离散空间分散方法。具体来说，在我们提出的空间分散方法中，为每棵草 (解决方案) 设计了一个随机插入邻域来生成新的种子，该过程有三个步骤。首先，d 作业从解决方案 (草) π 中随机选择并去除，不重复地形成一个部分序列 π_R 。剩余的作业在 π 中形成另一个部分序列 π' 。之后，根据 Wang 等人 (2011) 的研究，总处理时间较高的作业可能会导致其后续作业阻塞，从而产生比总处理时间较低的作业更大的阻塞时间。阻塞时间会增加最大完工时间 (makespan) 的值。总处理时间较低的作业应具有高优先级。因此，我们根据作业的总处理时间非递增顺序对 π_R 中的作业进行排序。最后，使用 NEH 启发式的插入方法，将 π_R 中的作业依次重新插入到 π' 中，直到重新建立完整的解决方案 π' 的 n 作业。对于每个作业的插入位置，评估 π' 的所有可能的插入位置，并选择最佳位置。

A numerical example of the random insertion neighborhood is shown in Fig. 1. The jobs 2 and 4 are firstly removed from $\pi = [1, 2, 3, 4, 5, 6]$ to construct two partial sequences $\pi_R = [2, 4]$ and $\pi' = [1, 3, 5, 6]$. Then, sort the jobs in π_R according to the non-decreasing of their total processing time, as a result, $\pi_R = [4, 2]$. Next, we try to insert the job 4 into each possible position of π' . The position with the lowest makespan (i.e., the position 2) is selected and the job 4 is inserted into it. Similarly, the job 2 is inserted into the position 5 of π' . Hence, a new seed $\pi' = [1, 4, 3, 5, 2, 6]$ is generated.

随机插入邻域的一个数值示例在图 1 中展示。首先从 $\pi = [1, 2, 3, 4, 5, 6]$ 中移除作业 2 和 4，以构建两个部分序列 $\pi_R = [2, 4]$ 和 $\pi' = [1, 3, 5, 6]$ 。然后，根据作业的总处理时间非递增顺序对 π_R 中的作业进行排序，结果为 $\pi_R = [4, 2]$ 。接下来，尝试将作业 4 插入 π' 的每个可能位置。选择具有最短最大完工时间 (即位置 2) 的位置，并将作业 4 插入其中。类似地，将作业 2 插入 π' 的位置 5。因此，生成了一个新的种子 $\pi' = [1, 4, 3, 5, 2, 6]$ 。

The number of the removed jobs, i.e., d , determines the performance of the discrete spatial dispersal. A larger $d\,(d \le n)$ can produce a larger neighborhood to explore the wider range, while a smaller d can be used to exploit the areas around the current solutions. Therefore, we draw support from the normal distribution and employ the proposed varying standard deviation $\sigma_{i,k}$ above to adjust d_i for each weed of the plant population. d_i is defined as:

被移除作业的数量，即 d ，决定了离散空间分散的性能。较大的 $d\,(d \le n)$ 可以产生更大的邻域以探索更广泛的范围，而较小的 d 可以用来开发当前解周围的区域。因此，我们从正态分布中获取支持，并使用上述提出的可变标准差 $\sigma_{i,k}$ 来调整每株植物群体的 d_i 。d_i 定义为:

$$d_i = \lfloor \left| N\left(0, \sigma_{i,k}^2\right) \right| \rfloor \tag{13}$$

where $N\left(0, \sigma_{i,k}^2\right)$ is a sampling function which returns a normally distributed random number with mean zero and the standard deviation $\sigma_{i,k}$. However, when a random number is sampled from the normal distribution, it may be close to the length of π or larger than it. If d_i is close to n , the proposed spatial dispersal would be simplified as NEH, which degrades its performance. Hence, d_i is restricted by the following way:

其中 $N\left(0, \sigma_{i,k}^2\right)$ 是一个采样函数，它返回均值为零且标准差为 $\sigma_{i,k}$ 的正态分布随机数。然而，当从正态分布中抽取一个随机数时，它可能接近 π 的长度或大于它。如果 d_i 接近 n ，则所提出的空间分散将简化为 NEH，这会降低其性能。因此，d_i 通过以下方式受到限制:

$$d_i = \lfloor \sigma_{\min} + \text{rand} \times (\sigma_{\max} - \sigma_{\min}) \rfloor \quad \text{if} \;\; d_i > \frac{n}{2} \text{ or } d_i < \sigma_{\min} \tag{14}$$

where rand is a uniformly random number between 0 and 1 . Eq. (14) makes d_i restrict in a reasonable area. $\sigma_{\min}$ and $\sigma_{\max}$ would be calibrated in Section 5.2.

其中 rand 是介于 0 和 1 之间的均匀随机数。等式 (14) 使得 d_i 限制在合理的区域内。$\sigma_{\min}$ 和 $\sigma_{\max}$ 将在 5.2 节中进行校准。

The detailed procedure of the proposed spatial dispersal is given in Algorithm 2. Note that lines 01-03 are the procedure of reproduction. The main computational burden in this spatial dispersal lies in lines 14-17, where all possible insertions have to be evaluated. The same speedup method as in the implementation of the PF-NEH(x)heuristic is employed to evaluate these insertions, so the generation of a new seed can be completed in $O(dm)$.

提出的空间分散详细过程在算法 2 中给出。注意，第 01-03 行是繁殖过程。这种空间分散的主要计算负担在于第 14-17 行，其中所有可能的插入都需要评估。采用与 PF-NEH(x) 启发式实现中相同的加速方法来评估这些插入，因此新种子的生成可以在 $O(dm)$ 内完成。

4.4. Local search

4.4. 局部搜索

To strengthen the local exploitation ability of the proposed DIWO algorithm, a local search is presented and embedded to quickly guide the individuals toward the best area. The referenced local search (RLS) proposed by Wang et al. (2010a) has been employed by many algorithms (Pan et al., 2013b; Tasgetiren et al., 2017; Wang et al., 2011) to solve the BFSP. In the RLS algorithm, $\pi_r = [\pi_r(1), \pi_r(2), \ldots, \pi_r(n)]$ is a reference sequence. The best solution so far is usually selected as the reference sequence. Let $\pi_c = [\pi_c(1), \pi_c(2), \ldots, \pi_c(n)]$ denote the current individual. The procedure of RLS can be briefly described as follows: (1) remove the job $\pi_r(i)$ from the current sequence π_c and test it in all positions of π_c ; (2) find the position with the lowest makespan and reinsert $\pi_r(i)$ into π_c to form a new individual π' ; (3) If π' is better than π_c , replace π_c with π' . Repeat (1) to (3) until all jobs in $\pi_r(i)$ are considered. In the RLS, the job positions of the current sequence π_c are assumed not be proper with respect to the reference sequence. The RLS algorithm will urge jobs to find better positions to be inserted in the current sequence π_c . However, this local search has its own shortcoming. If the reference sequence π_r is fixed, then the searching path is deterministic. In other words, the moving trajectories of all jobs are known. This case would cause the RLS to lack essential self-disturbance ability. If the reference sequence is not changed for many iterations, especially for later period of evolution, the population will be trapped in local optimum. Therefore, a shuffle-based referenced local search (SRLS) is proposed and embedded into DIWO for intensifying exploitation. In this local search, the first searching depends on the best solution so far. In the following search, the referenced sequence is shuffled, and the searching path is redefined. The procedure of SRLS is given in Algorithm 3.

为了增强所提出 DIWO 算法的本地开发能力，提出了一种局部搜索方法并将其嵌入，以快速引导个体向最佳区域移动。王等人 (2010a) 提出的参考局部搜索 (RLS) 已被许多算法 (Pan 等人，2013b；Tasgetiren 等人，2017；Wang 等人，2011) 用于解决 BFSP 问题。在 RLS 算法中，$\pi_r = [\pi_r(1), \pi_r(2), \ldots, \pi_r(n)]$ 是一个参考序列。到目前为止最好的解决方案通常被选为参考序列。令 $\pi_c = [\pi_c(1), \pi_c(2), \ldots, \pi_c(n)]$ 表示当前个体。RLS 的过程可以简述如下:(1) 从当前序列 π_c 中移除作业 $\pi_r(i)$ ，并在 π_c 的所有位置测试它; (2) 找到具有最短完工时间的位置，并将 $\pi_r(i)$ 重新插入 π_c 以形成新的个体 π' ; (3) 如果 π' 优于 π_c ，则用 π' 替换 π_c 。重复 (1) 到 (3)，直到考虑了 $\pi_r(i)$ 中的所有作业。在 RLS 中，假设当前序列 π_c 的作业位置相对于参考序列不适当。RLS 算法将促使作业找到更好的位置插入当前序列 π_c 中。然而，这种局部搜索有其自身的缺点。如果参考序列 π_r 是固定的，那么搜索路径是确定性的。换句话说，所有作业的移动轨迹都是已知的。这种情况会导致 RLS 缺乏必要的自我扰动能力。如果在许多迭代中，特别是在进化的后期阶段，参考序列没有改变，种群可能会陷入局部最优。因此，提出了一种基于洗牌的参考局部搜索

(SRLS) 并将其嵌入 DIWO 中以增强开发。在这种局部搜索中，第一次搜索依赖于到目前为止的最佳解决方案。在随后的搜索中，参考序列被打乱，搜索路径被重新定义。SRLS 的过程在算法 3 中给出。

```
Algorithm 2. Random-insertion-based spatial dispersal
算法 2. 基于随机插入的空间分散
Input: POP, $S_{\min}, S_{\max}, \sigma_{\min}$ and $\sigma_{\max}$
输入: POP, $S_{\min}, S_{\max}, \sigma_{\min}$ 和 $\sigma_{\max}$
Output: POP'
输出: POP'
1: for $i=1$ to $|POP|$ do
1: 对 $i=1$ 到 $|POP|$ 进行循环
Calculate $S_{\mathrm{i}}$ for $\pi_i$ in POP according to Eq. (10);
计算 POP 中的 $S_{\mathrm{i}}$ 对于 $\pi_i$ 的值，根据公式 (10);
: end for
: 循环结束
: for $i=1$ to $|\mathbf{POP}|$ do
: 对 $i=1$ 到 $|\mathbf{POP}|$ 进行循环
Calculate the $\sigma_{i,k}$ according to Eq. (12);
根据公式 (12) 计算 [̃ latex0̃];
for $j=1$ to $S_{\mathrm{i}}$ do
对 $j=1$ 到 $S_{\mathrm{i}}$ 进行循环
Generate a value of $d_i$ according to Eq. (13), and then check the legality of $d_i$ according to Eq. (14);
生成一个 $d_i$ 的值，根据公式 (13)，然后根据公式 (14) 检查 [̃ latex0̃] 的合法性;
$\pi_R \leftarrow \varnothing$ and $\pi' \leftarrow \varnothing$ ;
[̃ latex0̃] 和 [̃ latex1̃];
$\pi_R \leftarrow$ Randomly remove $d_i$ jobs from $\pi_i$ ;
$\pi_R \leftarrow$ 随机移除 $d_i$ 中的 $\pi_i$ 作业;
$\pi' \leftarrow \pi_i \setminus \pi_R$
$\pi_R \leftarrow$ Sort all jobs in $\pi_R$ with the non-decreasing sums of their processing time;
$\pi_R \leftarrow$ 将所有 $\pi_R$ 中的作业按照它们处理时间的总和进行非递减排序;
for $k=1$ to $d_i$ do
对于 $k=1$ 到 $d_i$ ;
Test the $k$ -th job of $\pi_R$ into all possible positions of $\pi'$ ;
测试 $k$ 的第 $\pi_R$ 项工作到 $\pi'$ 的所有可能位置;
Insert it into $\pi'$ at the position resulting in the lowest makespan;
将其插入到 $\pi'$ 中，位置使得最短完工时间 (makespan) 最低;
end for
结束循环;
$POP' \leftarrow POP' \cup \{\pi'\}$
end for
end for
end for
end for
```

In the local search above, shuffle (π_r) is a function to disturb the order of the jobs in π_r . The Fisher-Yates shuffle method (https://en.wikipedia.org/wiki/Fisher-Yates_shuffle) is used as the shuffle function. The variable flag ensures two different searching paths are performed in once local search. One is based on the best individual so far, the other is based on a random sequence. To avoid cycling search and getting trapped in local optimum, we perform the local search for each new seed created in the spatial dispersal procedure with a probability pls . For each new seed, a uniformly random number is generated in the interval [0, 1], if the random number is less than pls, the local search is performed on this new seed.

在上述局部搜索中，shuffle (π_r) 是一个用于打乱 π_r 中作业顺序的函数。使用 Fisher-Yates 洗牌方法 (https://en.wikipedia.org/wiki/Fisher-Yates_shuffle) 作为洗牌函数。变量 flag 确保在一次局部搜索中执行两条不同的搜索路径。一条基于迄今为止的最佳个体，另一条基于随机序列。为了避免循环搜索并陷入局部最优，我们以概率 pls 对在空间分散过程中创建的每个新种子执行局部搜索。对于每个新种子，在区间 [0, 1] 中生成一个均匀随机数，如果随机数小于 pls，则对该新种子执行局部搜索。

From the procedure of local search, it can be seen that the computational complexity is $O\left(2mn^3\right)$ in the worst case. The computational burden of SRLS is to evaluate each possible inserting position in lines 08-14 of Algorithm 3. Therefore, to reduce the computational time for evaluating them, two accelerating algorithms, i.e., the same speedup method used in PF-NEH(x)and the pruning procedure (Wang et al., 2012) are combined. To be specific, the speedup method is firstly employed to calculate the departure time $d'_{i,j}$ and the tail time $f'_{i,j}$ of job i on machine j in $\pi'.\pi'$ is the partial sequence by removing the job $\pi(a)$ from π . The tail time $f'_{i,j}$ is the duration between the latest starting time of job i on machine j and the end of the operation. Then, insert the job $\pi(a)$ into k th position of π' to form a new individual π'' and calculate the departure time $d''_{k,i}$ of $\pi(a)$ on each machine. Thereby, the makespan of π'' can be calculated by:

从局部搜索的过程可以看出，在最坏情况下计算复杂度是 $O\left(2mn^3\right)$ 。SRLS 的计算负担在于评估算法 3 的第 08-14 行中每个可能的插入位置。因此，为了减少评估它们的计算时间，结合了两种加速算法，即 PF-NEH(x) 中使用的相同加速方法和剪枝过程 (Wang et al., 2012)。具体来说，首先使用加速方法计算工作 i 在机器 j 上的出发时间 $d'_{i,j}$ 和尾部时间 $f'_{i,j}$ 。$\pi'.\pi'$ 是从 π 中移除工作 $\pi(a)$ 后的子序列。尾部时间 $f'_{i,j}$ 是工作 i 在机器 j 上的最晚开始时间与操作结束之间的持续时间。然后，将工作 $\pi(a)$ 插入到 π' 的第 k 个位置，形成一个新的个体 π'' ，并计算 $\pi(a)$ 在每台机器上的出发时间 $d''_{k,i}$ 。因此，π'' 的最大完工时间可以通过以下方式计算:

$$C_{\max}\left(\pi''\right)=\max_{j=1,2,\ldots,m}\left(d''_{k,j}+f'_{k,j}\right) \tag{15}$$

As regard the pruning procedure, when calculating the maximum value of $d''_{k,j}+f'_{k,j}$, if one of $d''_{k,j}+f'_{k,j}$ has been larger than the makespan of π , π'' must be not superior to π . So it is unnecessary to further evaluate π'' . π'' should be discarded and we can begin to consider the next insertion position.

关于剪枝过程，在计算 $d''_{k,j}+f'_{k,j}$ 的最大值时，如果其中一个 $d''_{k,j}+f'_{k,j}$ 已经大于 π 的最大完工时间，那么 π'' 必然不优于 π 。因此，没有必要进一步评估 π'' 。π'' 应该被舍弃，我们可以开始考虑下一个插入位置。

According to the speedup method, the computational complexity to evaluate an insertion operation can be reduced from $O(mn)$ to $O(m)$. The pruning procedure further reduces the computational effort. Thus, the computational complexity of the local search phase can be reduced to $O\left(2mn^2\right)$.

根据加速方法，评估一个插入操作的计算复杂度可以从 $O(mn)$ 降低到 $O(m)$ 。剪枝过程进一步减少了计算工作。因此，局部搜索阶段的计算复杂度可以降低到 $O\left(2mn^2\right)$ 。

4.5. Competitive exclusion

4.5. 竞争排除

The competitive exclusion determines which weeds can be allowed to pass to the next generation. In the basic IWO, all new created seeds and their parents are ranked together as a colony of weeds with respect to their fitness. Afterwards, weeds with lower fitness are eliminated to reach the maximum allowable population in a colony. However, this mechanism only considers the quality for the next plant population, and the diversity is ignored. If only select the best individuals for next iteration, the evolution of the population would stagnate soon. Therefore, a competitive exclusion proposed by Shao et al. (2017) is employed, in which a certain number of outstanding different individuals are selected as the population for next iteration. Firstly, a simple formula to check the similarity between two individuals is given as follows:

竞争排除决定了哪些杂草可以被允许传递到下一代。在基本的 IWO 中，所有新产生的种子及其亲本根据它们的适应性一起被排名，被视为一个杂草群落。之后，适应性较低的杂草被淘汰，以达到一个群落中允许的最大数量。然而，这种机制只考虑了下一代植物种群的质量，而忽略了多样性。如果仅选择最佳个体进行下一轮迭代，种群的进化将很快停滞。因此，采用了 Shao 等人 (2017 年) 提出的竞争排除方法，该方法选择了一定数量的杰出不同个体作为下一轮迭代的种群。首先，给出了一个检查两个个体之间相似度的简单公式如下:

$$\text{distance }\left(\pi_1,\pi_2\right)=\sum_{i=1}^{n} x_i\left(\pi_1(i),\pi_2(i)\right) \tag{16}$$

$$x_i\left(\pi_1(i),\pi_2(i)\right)=\begin{cases}0 & \pi_1(i)=\pi_2(i)\\ 1 & \pi_1(i)\neq\pi_2(i)\end{cases} \tag{17}$$

It can be seen from Eq. (16), π_1 is same as π_2 when distance $\left(\pi_1,\pi_2\right)=0$. Then, the new created seeds are combined with their parents to form a new plant population. The weeds in this new population are ranked according to their fitness, and then the former $P_{\max}$ outstanding weeds are reserved for next iteration. These reserved individuals are not similar to each other. The detailed implementation of the employed competitive exclusion is given in Algorithm 4.

从方程 (16) 可以看出，当距离 $(\pi_1, \pi_2) = 0$ 时，π_1 与 π_2 相同。然后，新产生的种子与它们的亲本结合，形成新的植物种群。这个新种群中的杂草根据它们的适应性进行排名，然后保留前 $P_{\max}$ 个杰出的杂草用于下一轮迭代。这些保留的个体之间互不相同。所采用的竞争排除的详细实现过程在算法 4 中给出。

It should be noted that the size of the initial plant population N_0 actually has less impact upon the final results of the algorithm since the number of weeds in the colony will quickly get the allowable maximum number of plants $P_{\max}$ (may be one iteration when the gap between $P_{\max}$ and N_0 is not large). For simplicity, we set N_0 to $P_{\max}$, which also reduces the number of parameters.

应该注意的是，初始植物种群的大小 N_0 实际上对算法的最终结果影响较小，因为群落中的杂草数量会迅速达到允许的最大植物数量 $P_{\max}$ (当 $P_{\max}$ 与 N_0 之间的差距不大时，可能只需一次迭代)。为了简化，我们设置 N_0 等于 $P_{\max}$ ，这也减少了参数的数量。

4.6. Overview of the DIWO algorithm

4.6. DIWO 算法概述

Having described each component of the proposed discrete invasive weed optimization algorithm, we summarize the procedure of it in Algorithm 5. In this algorithm, the PF-NEH(x)-based initialization

在描述了所提出离散入侵杂草优化算法的每个组成部分之后，我们在算法 5 中总结了其过程。在此算法中，基于 PF-NEH(x) 的初始化

Algorithm 3. Shuffle-based referenced local search Input: the individual π and the best individual $\pi_r = [\pi_r(1), \pi_r(2), \ldots, \pi_r(n)]$ so far

算法 3. 基于洗牌的参考局部搜索输入: 个体 π 和迄今为止的最佳个体 $\pi_r = [\pi_r(1), \pi_r(2), \ldots, \pi_r(n)]$

Output: π

输出: π

cnt $\leftarrow 1, j \leftarrow 0$, and flag $\leftarrow 0$;

计数器 cnt $\leftarrow 1, j \leftarrow 0$ ，和标志 flag $\leftarrow 0$;

Repeat

重复

while $cnt < n$ do

当 $cnt < n$ 时

$j \leftarrow \bmod (j+1, n)$;

$\pi' \leftarrow$ Remove the job $\pi_r(j)$ from π ;

$\pi' \leftarrow$ 从 π 中移除作业 $\pi_r(j)$;

bestfit $\leftarrow \infty$;

bestfit $\leftarrow \infty$;

for $i = 1$ to n do

对于 $i = 1$ 到 n 的每一个

$\pi'' \leftarrow$ Insert the job $\pi_r(j)$ into the position i of π' ;

$\pi'' \leftarrow$ 将作业 $\pi_r(j)$ 插入到 π' 的位置 i ;

if $C_{\max}(\pi'') <$ bestfit, then

如果 $C_{\max}(\pi'') <$ bestfit，那么

keypos $\leftarrow i$;

keypos $\leftarrow i$;

bestfit $\leftarrow C_{\max}(\pi'')$;

bestfit $\leftarrow C_{\max}(\pi'')$;

end if

结束 if

end for

结束 for

if $C_{\max}(\pi) >$ bestfit, then

如果 $C_{\max}(\pi) >$ bestfit，那么

$\pi \leftarrow$ Insert job $\pi_r(j)$ into the keypos-th position of π' ;

$\pi \leftarrow$ 将工作 $\pi_r(j)$ 插入到 π' 的 keypos-位置;

cnt $\leftarrow 1$;

cnt $\leftarrow 1$;

else

否则

```
cnt ← cnt +1 ;
cnt ← cnt +1 ;
end if
结束 if
end while
结束 while
if $flag == 0$ , then shuffle $(\pi_r)$ , and $flag \leftarrow flag + 1$ ;
如果 $flag == 0$ ，那么洗牌 $(\pi_r)$ ，并且 $flag \leftarrow flag + 1$ ;
3: Until $flag == 2$
3: 直到 $flag == 2$
```

Algorithm 4. Competitive exclusion

算法 4. 竞争排除

```
Input: $\mathbf{POP}, \mathbf{POP}'$ and $P_{\max}$
Output: POP
    $POP'' \leftarrow \varnothing$ ;
    $POP'' \leftarrow POP \cup POP'$;
 3: $\mathbf{POP}'' \leftarrow$ Sort the weeds in $\mathbf{POP}''$ with the increase of makespan;
   : $\mathbf{POP} \leftarrow \varnothing$ ;
    $j \leftarrow 2$ and $\mathbf{POP} \leftarrow \{\pi_1''\}$ , where $\pi_1'' \in \mathbf{POP}''$ ;
 : while $|\mathbf{POP}| < P_{\max}$ and $j \leq |\mathbf{POP}''|$ do
        flag ← true;
        for $i = 1$ to $|\mathbf{POP}|$ do
            if distance $(\pi_j'', \pi_i) == 0$ , where $\pi_j'' \in \mathbf{POP}''$ and $\pi_i \in \mathbf{POP}$ , then
                flag ← false;
            end if
        end for
        if $flag ==$ true, then $\mathbf{POP} \leftarrow \mathbf{POP} \cup \{\pi_j''\}$ ;
        $j \leftarrow j + 1$
 5: end while
```

method is employed to provide more promising candidate solutions. The random-insertion-based spatial dispersal is responsible for maintain the searching ability, the shuffle-based referenced local search is used to enhance the exploitation ability, and the competitive exclusion is used for constructing a good population for next iteration. These strategies enable DIWO to converge faster and provide high quality solutions.

采用该方法以提供更有前景的候选解。基于随机插入的空间分散负责维持搜索能力，基于洗牌的参考局部搜索用于增强开发能力，竞争排除用于构建良好的种群以供下一次迭代。这些策略使得 DIWO 更快收敛并提供高质量的解。

5. Numerical experiments

5. 数值实验

In this section, an extensive experimental evaluation is carried out to investigate the performance of the presented algorithm. First of all, the test instances and the experiment settings are presented in Section 5.1. Then, the experimental results are reported according to the following aspects:

在本节中，进行了广泛的实验评估以研究所提算法的性能。首先，在第 5.1 节中介绍了测试实例和实验设置。然后，根据以下方面报告实验结果:

1. Calibration of parameters.
1. 参数校准。
2. Assessment of the effectiveness of the proposed spatial dispersal.
2. 评估所提空间分散的有效性。
3. Assessment of the effectiveness of the local search.
3. 评估局部搜索的有效性。

4. Comparison of the proposed algorithm with the existing well-performing algorithms.
4. 将所提算法与现有表现良好的算法进行比较。

```
Algorithm 5. DIWO algorithm
算法 5. DIWO 算法
Input: $P_{\max}, S_{\min}, S_{\max}, \sigma_{\min}, \sigma_{\max}$ and pls.
输入: $P_{\max}, S_{\min}, S_{\max}, \sigma_{\min}, \sigma_{\max}$ 和请。
Output: The best one in POP
输出: 在 POP 中的最佳个体
01: Initialize the plant population POP (described in Algorithm 1);
01: 初始化植物种群 POP(在算法 1 中描述);
$k \leftarrow 1$
3: repeat
3: 重复
% Reproduction (described lines 01-03 in Algorithm 2)
% 繁殖 (在算法 2 的第 01-03 行中描述)
Calculate the number of seeds generated by each weed in POP;
计算 POP 中每株杂草产生的种子数量;
$POP' \leftarrow \varnothing$ ;
for $i = 1$ to $P_{\max}$ do  % Spatial dispersal (described in lines 04-21 Algorithm 2)
对于 $i = 1$ 到 $P_{\max}$ 执行  % 空间扩散 (在算法 2 的第 04-21 行中描述)
Calculate $\sigma_{i,k}$ for the $i$ -th weed of $\mathbf{POP}$ ;
计算 $\sigma_{i,k}$ 对于 $i$ -th 杂草的 $\mathbf{POP}$ ;
Generate $S_{\mathrm{i}}$ new seeds for the $i$ -th weed of $\mathbf{POP}$ ;
为 $\mathbf{POP}$ 中的第 $i$ -th 杂草生成 $S_{\mathrm{i}}$ 新种子;
Save these $S_{\mathrm{i}}$ new seeds into $\mathbf{POP}'$ ;
将这些 $S_{\mathrm{i}}$ 新种子保存到 $\mathbf{POP}'$ 中;
end for
结束循环
for $i = 1$ to $\left|\mathbf{POP}'\right|$ do $\mid$ do
对于 $i = 1$ 到 $\left|\mathbf{POP}'\right|$ do $\mid$ 的每一个步骤
if $rand < pls$ , then
如果 $rand < pls$ , 那么
Perform SRLS on the $i$ -th individual in $\mathbf{POP}'$ ;
在 $\mathbf{POP}'$ 中的第 $i$ 个个体上执行 SRLS;
end if
结束如果
end for
结束循环
% Competitive exclusion (described in Algorithm 4)
% 竞争排除 (在算法 4 中描述)
Select $P_{\max}$ different outstanding individuals from $\mathbf{POP}$ and $\mathbf{POP}'$ to construct a new plant population $\mathbf{POP}$
从 $\mathbf{POP}$ 和 $\mathbf{POP}'$ 中选择 $P_{\max}$ 个不同的杰出个体构建一个新的植物种群 $\mathbf{POP}$
for next iteration;
用于下一次迭代;
$k \leftarrow k + 1$
16: until the termination criterion is met
16: 直到满足终止条件
```

5.1. Experiment settings

5.1. 实验设置

To investigate the performance of the DIWO, the benchmark set introduced by Taillard (1993) is adopted. The Taillard's benchmark set consists of 120 instances, ranging from 20 jobs and 5 machines to 500 jobs and 20 machines. These instances are divided into 12 subsets, each of which consists of 10 instances with the same size. Since the BFSP does not have a tight lower bound which is used to compare the results obtained by several algorithms, we adopt best known solutions instead of the lower bound values. Note that calibrating algorithms with the same instances that will later be used for computational results and comparisons constitutes poor practices, which would lead to biased or over-fitting results (Pan and Ruiz, 2014). Therefore, to avoid over-fitting in the results, we randomly regenerate a new test set with $n \in \{40, 80, 120, 160, 200\}$ and $m \in \{5, 10, 15, 20, 25\}$ through Taillard' s generation method (Taillard,1993). Each combination of n and m has five instance, resulting in a total of 125 instances. 125 instances are enough for calibration. The processing time is uniformly distributed from 1 to 99 . These test instances are employed to calibrate the parameters of the proposed algorithm.

为了研究 DIWO 的性能，采用了 Taillard (1993) 引入的基准测试集。Taillard 的基准测试集包含 120 个实例，范围从 20 个工作项和 5 台机器到 500 个工作项和 20 台机器。这些实例被分为 12 个子集，每个子集包含 10 个大小相同的实例。由于 BFSP 没有紧的下界，该下界用于比较几种算法得到的结果，因此我们采用最佳已知解而不是下界值。请注意，使用稍后用于计算结果和比较的相同实例来校准算法构成不良实践，这可能导致有偏见或过拟合的结果 (Pan 和 Ruiz，2014)。因此，为了避免结果过拟合，我们通过 Taillard 的生成方法 (Taillard，1993) 随机重新生成一个包含 $n \in \{40, 80, 120, 160, 200\}$ 和 $m \in \{5, 10, 15, 20, 25\}$ 的新测试集。每种 n 和 m 的组合都有五个实例，总共产生 125 个实例。125 个实例足以进行校准。处理时间均匀分布在 1 到 99 之间。这些测试实例被用于校准所提出算法的参数。

In this study, all algorithms are coded in Java language and executed on a personal computer that has an Intel Core (TM) CPU with 3.20GHz , 4G RAM, and Windows 7 Operating System. The termination condition is set as the maximum elapsed CPU time $t = \rho$mn milliseconds, where ρ is set to different values in the following sections. To analyze the experimental results obtained, the Average Relative Percent Deviation (ARPD) is collected to measure the average relative quality of the solutions, which can be calculated as follows:

在本研究中，所有算法均使用 Java 语言编写，并在一台配备了 Intel Core(TM)CPU 3.20GHz，4G 内存和 Windows 7 操作系统的个人计算机上执行。终止条件设置为最大的 CPU 经过时间 $t = \rho$mn 毫秒，其中 ρ 在以下各节中设置为不同的值。为了分析获得的实验结果，收集平均相对百分偏差 (ARPD) 来衡量解决方案的平均相对质量，计算方式如下:

$$\mathrm{ARPD} = \frac{1}{R} \sum_{i=1}^{R} \frac{C_i - C_{\min}}{C_{\min}} \times 100 \tag{18}$$

where R is the number of runs. C_i is the solution generated by a specific algorithm in the i th experiment for a given instance. As mentioned above, two different types of instance are employed in this study. The first one is the Taillard's instances, which are used for the comparison of algorithms in the literature. Ribas et al. (2011) have reported the best known solutions (makespan) for these instances, so we set them as $C_{\min}$. The second one is the 125 new instances generated by ourselves, which are used for parameters calibration. These instances do not have the best known solutions. Thus, the minimum makespan found by all algorithms is selected as $C_{\min}$. Obviously, the smaller the value of ARPD is, the better the performance of the algorithm is.

其中 R 是运行的次数。C_i 是特定算法在 i 次实验中对给定实例生成的解决方案。如上所述，本研究使用了两种不同类型的实例。第一种是 Taillard 的实例，用于文献中算法的比较。Ribas 等人 (2011 年) 已经报告了这些实例的最佳已知解决方案 (完工时间)，因此我们将它们设置为 $C_{\min}$ 。第二种是我们自己生成的 125 个新实例，用于参数校准。这些实例没有最佳已知解决方案。因此，选择所有算法找到的最小完工时间作为 $C_{\min}$ 。显然，ARPD 的值越小，算法的性能越好。

5.2. Design of experiments for the calibration of parameters

5.2. 参数校准的实验设计

The selection of parameters has a significant influence on the effectiveness and efficiency of stochastic algorithms. Usually, the parameters of an algorithm are determined through two ways: the experience in the previous literature and experimental way. For the first way, we can set parameters through summarizing previous relevant literature. Especially for similar problems, the values of parameters in the previous literature have a strong instructive significance, which can further help us narrow the scope of the selection. However, this way often cannot provide accurate values. For the second way, it obtains the optimal calibration through testing many potential values of parameters. Meanwhile, it also analyzes the sensitivity of parameters by using several statistical methods. But it often consumes considerable computational time due to test all combinations of parameters, and its accuracy is influenced by the potential values of parameters. Therefore, in our parameters calibration, the merits of these two ways are combined. To be specific, firstly, the selection range of each parameter for DIWO is determined according to the previous literature (Sang et al., 2018a, b; Shao et al., 2017) and some preliminary experiments. Several levels (values) for a factor (parameter) are determined by trial and error. Then, in order to find the best levels for these factors, the design of experiments (DOE) approach (Montgomery, 2008) is employed. Each combination would be tested by using some additional instances. The multi-factor analysis of variance (ANOVA) procedure would be used to analyze the obtained experimental results to determine the main effects and interaction of parameters, and then establish the best combination of parameters.

参数的选择对随机算法的有效性和效率具有重要影响。通常，算法的参数通过两种方式确定: 以往文献中的经验和实验方法。对于第一种方式，我们可以通过总结之前的相关文献来设置参数。特别是对于类似的问题，以往文献中参数的值具有强烈的指导意义，这可以进一步帮助我们缩小选择范围。然而，这种方式通常无法提供准确的值。对于第二种方式，它通过测试许多潜在的参数值来获得最优校准。同时，它还使用几种统计方法来分析参数的敏感性。但由于测试所有参数组合，这种方式通常消耗大量的计算时间，其准确性受到参数潜在值的影响。因此，在我们的参数校准中，结合了这两种方式的优点。具体来说，首先，根据以往文献 (Sang 等人，2018a, b；Shao 等人，2017) 和一些初步实验，确定了 DIWO 中每个参数的选择范围。通过试错确定了一个因素 (参数) 的几个水平 (值)。然后，为了找到这些因素的最佳水平，采用了实验设计 (DOE) 方法 (Montgomery，2008)。每种组合都将使用一些额外实例进行测试。多因素方差分析 (ANOVA) 过程将用于分析获得的实验结果，以确定参数的主要效应和交互作用，然后建立最佳参数组合。

Table 1

表 1

ANOVA results over calibrating the parameters of DIWO.

DIWO 参数校准的 ANOVA 结果。

Source	Sum of squares	Degrees of freedom	Mean Square	F -ratio	p -value
$P_{\max}$	91.01	2	45.5038	321.42	0.0000
$S_{\min}$	45.04	2	22.5224	159.09	0.0000
$S_{\max}$	2.71	2	1.3554	9.57	0.0001
$\sigma_{\min}$	11.80	2	5.9012	41.68	0.0000
$\sigma_{\max}$	0.21	2	0.1055	0.75	0.4746
pls	9.88	2	4.9415	34.90	0.0000
$P_{\max} * S_{\min}$	33.96	4	8.4898	59.97	0.0000
$P_{\max} * S_{\max}$	2.50	4	0.6250	4.41	0.0014
$P_{\max} * \sigma_{\min}$	0.78	4	0.1959	1.38	0.2368
$P_{\max} * \sigma_{\max}$	0.87	4	0.2168	1.53	0.1899
$P_{\max} * pls$	22.56	4	6.2901	44.43	0.0000
$S_{\min} * S_{\max}$	2.28	4	0.5697	4.02	0.0029
$S_{\min} * \sigma_{\min}$	0.66	4	0.1661	1.17	0.3204
$S_{\min} * \sigma_{\max}$	1.22	4	0.3042	2.15	0.0720
$S_{\min} * pls$	6.20	4	1.5494	10.94	0.0000
$S_{\max} * \sigma_{\min}$	10.87	4	2.7184	19.20	0.0000
$S_{\max} * \sigma_{\max}$	0.62	4	0.1557	1.10	0.3548
$S_{\max} * pls$	0.30	4	0.0761	0.54	0.7082
$\sigma_{\min} * \sigma_{\max}$	0.79	4	0.1964	1.39	0.2355
$\sigma_{\min} * pls$	0.37	4	0.0932	0.66	0.6209
$\sigma_{\max} * pls$	0.18	4	0.0459	0.32	0.8620
Residual	8246.11	58 247	0.1416		
Total	8493.54	58 319			

来源	平方和	自由度	均方	F -比率	p -值
$P_{\max}$	91.01	2	45.5038	321.42	0.0000
$S_{\min}$	45.04	2	22.5224	159.09	0.0000
$S_{\max}$	2.71	2	1.3554	9.57	0.0001
$\sigma_{\min}$	11.80	2	5.9012	41.68	0.0000
$\sigma_{\max}$	0.21	2	0.1055	0.75	0.4746
pls(偏最小二乘回归)	9.88	2	4.9415	34.90	0.0000
$P_{\max} * S_{\min}$	33.96	4	8.4898	59.97	0.0000
$P_{\max} * S_{\max}$	2.50	4	0.6250	4.41	0.0014
$P_{\max} * \sigma_{\min}$	0.78	4	0.1959	1.38	0.2368
$P_{\max} * \sigma_{\max}$	0.87	4	0.2168	1.53	0.1899
$P_{\max} * pls$	22.56	4	6.2901	44.43	0.0000
$S_{\min} * S_{\max}$	2.28	4	0.5697	4.02	0.0029
$S_{\min} * \sigma_{\min}$	0.66	4	0.1661	1.17	0.3204
$S_{\min} * \sigma_{\max}$	1.22	4	0.3042	2.15	0.0720
$S_{\min} * pls$	6.20	4	1.5494	10.94	0.0000
$S_{\max} * \sigma_{\min}$	10.87	4	2.7184	19.20	0.0000
$S_{\max} * \sigma_{\max}$	0.62	4	0.1557	1.10	0.3548
$S_{\max} * pls$	0.30	4	0.0761	0.54	0.7082
$\sigma_{\min} * \sigma_{\max}$	0.79	4	0.1964	1.39	0.2355
$\sigma_{\min} * pls$	0.37	4	0.0932	0.66	0.6209
$\sigma_{\max} * pls$	0.18	4	0.0459	0.32	0.8620
残差	8246.11	58 247	0.1416		
总计	8493.54	58 319			

From the procedure of the DIWO algorithm, there are six key parameters: the allowable maximum number of plants $(P_{\max})$, the minimum number of seeds $(S_{\min})$, the maximum number of seeds $(S_{\max})$, the minimum value of standard deviation $(\sigma_{\min})$, the maximum value of standard deviation $(\sigma_{\max})$ and the probability of the local search (pls). The levels for each factor are set as follows: $P_{\max} \in \{10, 30, 50\}$, $S_{\min} \in \{0, 1, 2\}, S_{\max} \in \{5, 7, 9\}, \sigma_{\min} \in \{3, 4, 5\}, \sigma_{\max} \in \{8, 9, 10\}$, and $pls \in \{0.15, 0.25, 0.45\}$. These parameters above yield a total of $3 \times 3 \times 3 \times 3 \times 3 \times 3 = 729$ different configurations for the DIWO algorithm. Then, we employ DOE approach to study the influence of parameters on the performance of the DIWO algorithm. Each configuration is executed 5 times for each instance. The termination criterion is set $\rho = 30$. As a result, a total of $729 \times 5 \times 125 = 455625$ results are generated in this experiment. The total CPU time for calibration is almost 284.76 CPU days. Due to multi-cores in our personal computer, actually we only used almost 9.5 days to complete the whole experiment.

从 DIWO 算法的过程来看，有六个关键参数: 允许的最大植物数量 $(P_{\max})$ ，最小种子数量 $(S_{\min})$ ，最大种子数量 $(S_{\max})$ ，标准差的最小值 $(\sigma_{\min})$ ，标准差的最大值 $(\sigma_{\max})$ 以及局部搜索的概率 (pls)。每个因素的水平设置如下: $P_{\max} \in \{10, 30, 50\}$ ， $S_{\min} \in \{0, 1, 2\}, S_{\max} \in \{5, 7, 9\}, \sigma_{\min} \in \{3, 4, 5\}, \sigma_{\max} \in \{8, 9, 10\}$ 和 $pls \in \{0.15, 0.25, 0.45\}$ 。上述参数为 DIWO 算法提供了 $3 \times 3 \times 3 \times 3 \times 3 \times 3 = 729$ 种不同的配置。然后，我们采用 DOE 方法研究参数对 DIWO 算法性能的影响。每种配置在每个实例上执行 5 次。终止准则设置为 $\rho = 30$ 。因此，在这个实验中总共生成了 $729 \times 5 \times 125 = 455625$ 个结果。校准的总 CPU 时间几乎是 284.76 CPU 天。由于我们的个人电脑具有多核，实际上我们只用了近 9.5 天来完成整个实验。

The experimental results are analyzed by means of ANOVA. In the experiment, three main hypotheses, i.e., normality, homogeneity of variance and independent of the residuals, are checked and accepted. Note that we focus on the F -ratio which is a clear indicator of significance when the p -values are less than the confidence level. The larger the F -ratio is, the more effect the factor has on the response variable. Moreover, we do not consider the interactions more than two factors since their F -ratios are very small. The ANOVA results are reported in Table 1.

实验结果通过 ANOVA 方法进行分析。在实验中，检查并接受了三个主要假设，即残差的正态性、方差同质性和独立性。请注意，我们关注的是 F -比率，当 p -值小于置信水平时，它是显著性的明确指标。F -比率越大，该因素对响应变量的影响越大。此外，由于两个以上因素的 F -比率非常小，我们不考虑两个以上因素的交互作用。ANOVA 结果在表 1 中报告。

As seen in Table 1, the p -values of parameter $P_{\max}, \sigma_{\min}, S_{\min}, pls$ and $S_{\max}$ are less than $\alpha = 0.05$ confidence level, which indicates that these factors are significant and influence the performance of the DIWO algorithm. The allowable maximum number of plants $(P_{\max})$ achieves the largest F -ratio, which indicates that $P_{\max}$ is the significant factor that affects the performance of the proposed DIWO. Fig. 2 gives the main effects plot of all parameters. It is clearly observed that the choice of $P_{\max} = 10$ obtains the best result, while $P_{\max} = 30$ yields the worst result. It demonstrates that a larger population increases the computational time in once iteration and reduce the convergence speed which are not desirable. The second largest F -ratio value corresponds to the factor $S_{\min}$. It can be observed from Fig. 2 that the value $S_{\min} = 0$ yields the best result, while $S_{\min} = 2$ obtains the worst result. It can be explained that a larger $S_{\min}$ may enlarge the number of offspring individuals with lower fitness in each generation, which would decrease the quality of the whole population. The next greatest F -ratio corresponds to pls. It can be seen that $pls = 0.45$ or 0.25 provides the worse results, while $pls = 0.15$ obtains the best result. It demonstrates that the larger probability would lead the population to trap into local optimum. The fourth F -ratio value is relative to $\sigma_{\min}$. As seen in Fig. $2, \sigma_{\min} = 5$ achieves the best result, while $\sigma_{\min} = 3$ yields the worst result. Since $\sigma_{\min}$ determines the smallest searching range, a lower value of $\sigma_{\min}$ would lead to inadequate searching. The last significant factor is $S_{\max}$. It is clear from Fig. 2 that too small or too large $S_{\max}$ would lead to deterioration of the algorithm’ s performance, while $S_{\max} = 7$ gives the best average performance.

如表 1 所示，参数 p 的 $P_{\max}, \sigma_{\min}, S_{\min}, pls$ 值和 $S_{\max}$ 值小于 $\alpha = 0.05$ 置信水平，这表明这些因素是显著的，并影响 DIWO 算法的性能。允许的最大植物数量 $(P_{\max})$ 达到最大的 F 比率，这表明 $P_{\max}$ 是影响所提出 DIWO 性能的显著因素。图 2 给出了所有参数的主要效应图。可以明显看出，选择 $P_{\max} = 10$ 会得到最佳结果，而 $P_{\max} = 30$ 则得到最差结果。这表明较大的人口会增加单次迭代的计算时间并降低收敛速度，这是不希望看到的。第二大的 F 比率值对应于因素 $S_{\min}$ 。从图 2 可以看出，值 $S_{\min} = 0$ 得到最佳结果，而 $S_{\min} = 2$ 得到最差结果。可以这样解释，较大的 $S_{\min}$ 可能会增加每一代中低适应度后代个体的数量，这会降低整个种群的素质。下一个最大的 F 比率对应于 pls。可以看出 $pls = 0.45$ 或 0.25 提供较差的结果，而 $pls = 0.15$ 得到最佳结果。这表明较大的概率会导致种群陷入局部最优。第四个 F 比率值与 $\sigma_{\min}$ 相关。如图所示 $2, \sigma_{\min} = 5$ 得到最佳结果，而 $\sigma_{\min} = 3$ 得到最差结果。由于 $\sigma_{\min}$ 决定了最小的搜索范围，较低的 $\sigma_{\min}$ 值将导致搜索不足。最后一个显著因素是 $S_{\max}$ 。从图 2 可以看出，过小或过大的 $S_{\max}$ 会导致算法性能下降，而 $S_{\max} = 7$ 则提供了最佳的平均性能。

However, if there are significant interactions between parameters, it is not meaningful to consider the value of a single parameter. Table 1 also displays the 2-level interactions of all parameters. It is observed from it that the interactions $P_{\max} * S_{\min}, P_{\max} * S_{\max}, P_{\max} * pls$, $S_{\max} * S_{\min}, S_{\min} * \sigma_{\min}$ and $S_{\min} * pls$ are significant since the p - values are less than 0.05 . Thus, we have to consider the interactions of these parameters. Fig. 3 shows the interaction plot of these parameters. Although many interactions are statistically significant, all interactions except for $S_{\max} * S_{\min}$ are weak according to Fig. 3 and they do not influence the conclusions reached from Fig. 2. For the interaction $S_{\max} * S_{\min}, S_{\min} = 0$ generates the lowest ARPD with $S_{\max} = 5$, which is different from Fig. 2. Nonetheless, from other interactions related to $S_{\max}$, we can see that $S_{\max} = 7$ yields the good results. Moreover, the F -ratio of $S_{\max} * S_{\min}$ is relatively smaller than other factors. Thus, the interaction $S_{\max} * S_{\min}$ is also weak, and $S_{\max}$ should be set to 7 .

然而，如果参数之间存在显著相互作用，那么考虑单个参数的值是没有意义的。表 1 也显示了所有参数的 2 级相互作用。从表中可以看出，相互作用 $P_{\max} * S_{\min}, P_{\max} * S_{\max}, P_{\max} * pls$, $S_{\max} * S_{\min}, S_{\min} * \sigma_{\min}$ 和 $S_{\min} * pls$ 是显著的，因为 p -值小于 0.05。因此，我们必须考虑这些参数的相互作用。图 3 展示了这些参数的相互作用图。尽管许多相互作用在统计学上显著，但根据图 3，除了 $S_{\max} * S_{\min}$ 之外的所有相互作用都是微弱的，它们并不影响从图 2 得出的结论。对于相互作用 $S_{\max} * S_{\min}, S_{\min} = 0$ ，它在与 $S_{\max} = 5$ 结合时产生最低的 ARPD，这与图 2 不同。尽管如此，从与其他与 $S_{\max}$ 相关的相互作用中，我们可以看到 $S_{\max} = 7$ 产生了良好的结果。此外，$S_{\max} * S_{\min}$ 的 F -比率相对于其他因素来说较小。因此，相互作用 $S_{\max} * S_{\min}$ 也是微弱的，$S_{\max}$ 应该设置为 7。

However, although Fig. 2 has shown the best value for each parameter, we cannot confirm that whether lower values for $P_{\max}$ and $S_{\max}$, and larger values for $\sigma_{\min}$ can obtain better performance. Therefore, an additional experiment is conducted to find the best values for them. In this experiment, we set $P_{\max} \leq 10$, since 10 is the current best value for $P_{\max}$ in the previous experiment. The potential values of $P_{\max}$ are set $\{4,6,8,10\}$. Similarly, we set $pls \leq 0.15$, i.e. $pls \in \{0.05, 0.10, 0.15\}$. Moreover, since $5 \leq \sigma_{\min} \leq \sigma_{\max}$ and the algorithm achieves the best performance when $\sigma_{\max} = 10$, the potential values of $\sigma_{\min}$ are included in $\{5,7,9,10\}$. For other three parameters, i.e., $S_{\max}, S_{\min}$ and $\sigma_{\max}$, we set them according to Fig. 2. Thus, there are a total of $4 \times 3 \times 4 = 48$ different configurations. The new 125 instances generated in Section 5.1 are also used as the test bed. Each configuration is run 5 times for each instance. The termination criterion is still set $\rho = 30$. As a result, a total of $48 \times 5 \times 125 = 30000$ results are generated in this experiment. The results above are also analyzed by the ANOVA technique. The analysis results are reported here in the form of ANOVA charts with 95% confidence intervals in Fig. 4. Note that overlapping confidence intervals signify that the observed differences in the response variables are not statistically significant. In other words, if the confidence intervals of two potential values for a parameter are overlapped, these two potential values cannot bring the significant changes for the algorithm.

然而，尽管图 2 显示了每个参数的最佳值，我们无法确认 $P_{\max}$ 和 $S_{\max}$ 的较小值以及 $\sigma_{\min}$ 的较大值是否能获得更好的性能。因此，进行了额外的实验以找到它们的最佳值。在这个实验中，我们设置 $P_{\max} \leq 10$ ，因为在前一个实验中，10 是 $P_{\max}$ 的当前最佳值。$P_{\max}$ 的潜在值设置为 $\{4,6,8,10\}$ 。类似地，我们设置 $pls \leq 0.15$ ，即 $pls \in \{0.05, 0.10, 0.15\}$ 。此外，由于 $5 \leq \sigma_{\min} \leq \sigma_{\max}$ ，且当 $\sigma_{\max} = 10$ 时算法达到最佳性能，$\sigma_{\min}$ 的潜在值包括在 $\{5,7,9,10\}$ 中。对于其他三个参数，即 $S_{\max}, S_{\min}$ 和 $\sigma_{\max}$ ，我们根据图 2 设置它们。因此，总共有 $4 \times 3 \times 4 = 48$ 种不同的配置。在第 5.1 节中生成的新 125 个实例也用作测试

平台。每种配置对每个实例运行 5 次。终止标准仍然设置为 $\rho = 30$ 。因此，在这个实验中总共生成了 $48 \times 5 \times 125 = 30000$ 个结果。上述结果也通过 ANOVA 技术进行了分析。分析结果以带有 95% 置信区间的 ANOVA 图表的形式报告在图 4 中。请注意，重叠的置信区间意味着观测到的响应变量之间的差异在统计学上不显著。换句话说，如果一个参数的两个潜在值的置信区间重叠，那么这两个潜在值不能为算法带来显著的变化。

As can be seen from Fig. 4, when $P_{\text{max}} = 10$ and $S_{\text{max}} = 0.15$, DIWO can achieve the best performance, which is in consistent with Fig. 2. Meanwhile, the confidence intervals of $P_{\text{max}} = 10$ and $S_{\text{max}} = 0.15$ are not overlapped with other potential values, which indicates $P_{\text{max}} = 10$ and $S_{\text{max}} = 0.15$ are the best selections for DIWO. Additionally, we can also conclude that lower P_{max} and S_{max} cannot enable DIWO to obtain better performance. However, σ_{min} has a different behavior. It can be seen from Fig. 4 that the performance of DIWO on $\sigma_{\text{min}} = 7$ and $\sigma_{\text{min}} = 10$ is better than on $\sigma_{\text{min}} = 7$ and $\sigma_{\text{min}} = 9$. But these differences are not significant, since the confidence intervals of these potential values are overlapped. In other words, the larger σ_{min} cannot bring the significant enhancement for DIWO, and σ_{min} can be set to a value between 5 and

如图 4 所示, 当 $P_{\text{max}} = 10$ 和 $S_{\text{max}} = 0.15$ 时, DIWO 能够达到最佳性能, 这与图 2 一致。同时, $P_{\text{max}} = 10$ 和 $S_{\text{max}} = 0.15$ 的置信区间与其他潜在值不重叠，这表明 $P_{\text{max}} = 10$ 和 $S_{\text{max}} = 0.15$ 是 DIWO 的最佳选择。此外，我们还可以得出结论，较低的 P_{max} 和 S_{max} 不能使 DIWO 获得更好的性能。然而，σ_{min} 的行为有所不同。从图 4 可以看出，DIWO 在 $\sigma_{\text{min}} = 7$ 和 $\sigma_{\text{min}} = 10$ 上的性能优于 $\sigma_{\text{min}} = 7$ 和 $\sigma_{\text{min}} = 9$ 。但这些差异并不显著，因为这些潜在值的置信区间是重叠的。换句话说，更大的 σ_{min} 不能为 DIWO 带来显著的提升，σ_{min} 可以设置为 5 到某个值之间。10.

On the basis of the analysis above, the parameters are recommended as follows: $P_{\text{max}} = 10, S_{\text{min}} = 0, S_{\text{max}} = 7, \sigma_{\text{min}} = 0, \sigma_{\text{max}} = 5$, and pls $= 0.15$.

在上述分析的基础上，建议参数设置如下: $P_{\text{max}} = 10, S_{\text{min}} = 0, S_{\text{max}} = 7, \sigma_{\text{min}} = 0, \sigma_{\text{max}} = 5$ ，以及 pls $= 0.15$ 。

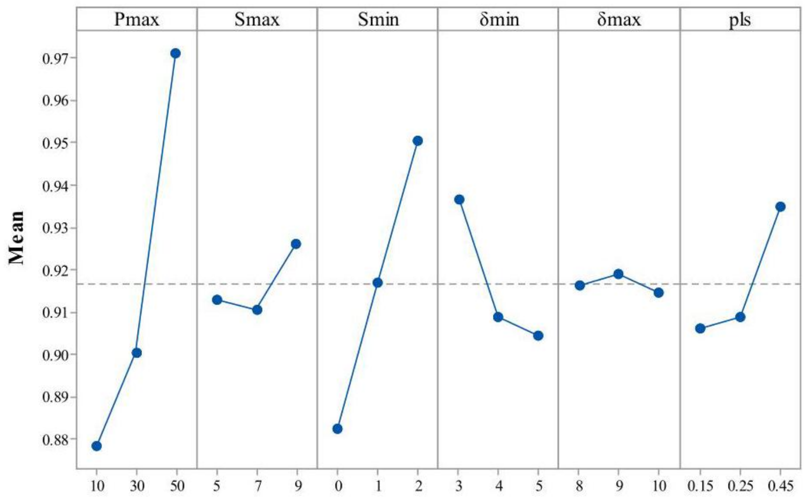

Fig. 2. Main effects plot of parameters.

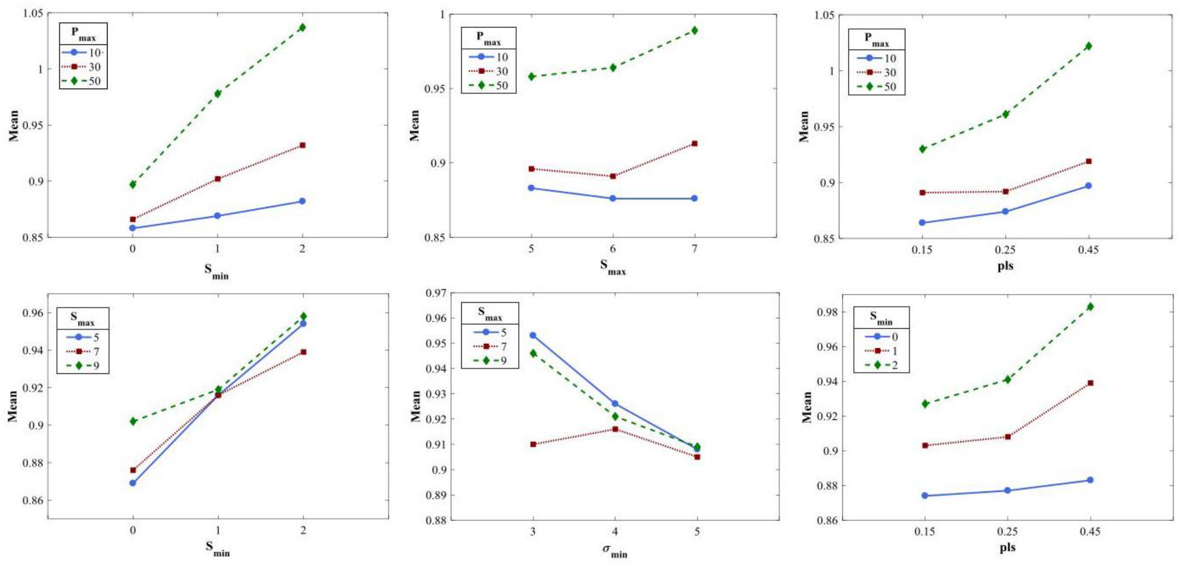

Fig. 3. Interaction plot of parameters.

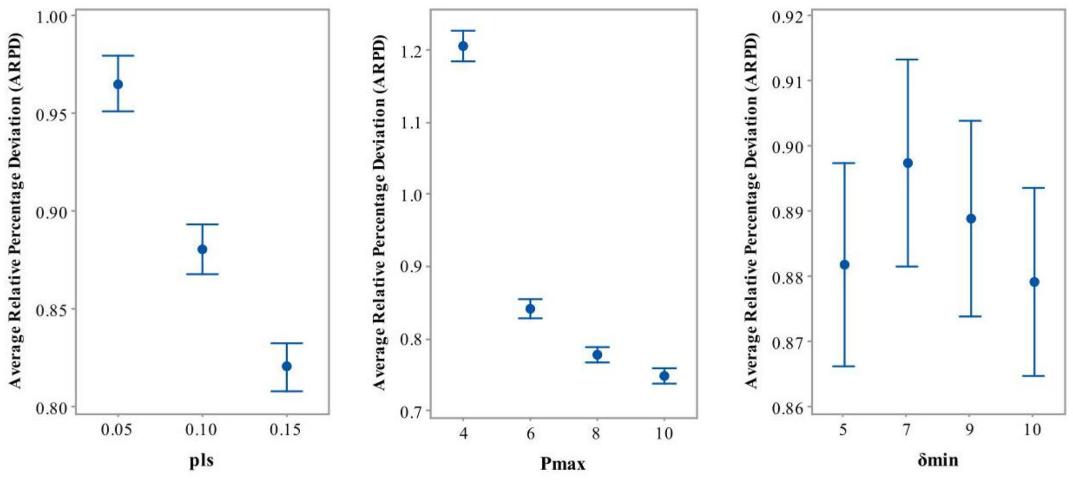

Fig. 4. Means plot and 95% confidence level intervals of $P_{\max}, S_{\max}$ and $\sigma_{\min}$.
图 4。平均值图和 95% 的 $P_{\max}, S_{\max}$ 和 $\sigma_{\min}$ 置信水平区间。
Table 2

表 2
The ARPD of SIWO_NL, SDIWO, DIWO_NL, SaDIWO_NS, SaDIWO and DIWO.
SIWO_NL、SDIWO、DIWO_NL、SaDIWO_NS、SaDIWO 和 DIWO 的 ARPD。

Instance	SDIWO_NL	SDIWO	DIWO_NL	SaDIWO_NS	SaDIWO	DIWO
20×5	2.396	0.008	0.078	0.012	0.017	0.000
20×10	2.556	0.007	0.046	0.013	0.008	0.000
20×20	1.727	0.000	0.019	0.010	0.010	0.000
50×5	3.351	0.450	0.374	0.028	0.021	0.128
50×10	3.572	0.228	0.224	0.072	0.105	-0.073
50×20	5.090	0.231	0.521	0.248	0.312	0.086
100×5	1.405	-0.218	-0.587	-0.776	-0.746	-0.775
100×10	1.764	-0.582	-0.652	-0.672	-0.657	-0.972
100×20	2.922	-0.336	-0.173	-0.119	-0.139	-0.741
200×10	0.452	-0.809	-1.021	-0.759	-0.763	-1.200
200×20	0.500	-1.245	-1.296	-0.917	-0.889	-1.591
500×20	-1.682	-2.594	-2.787	-2.438	-2.463	-2.835
Average	2.005	-0.405	-0.438	-0.441	-0.432	-0.664

实例	SDIWO_NL	SDIWO	DIWO_NL	SaDIWO_NS	SaDIWO	DIWO
20×5	2.396	0.008	0.078	0.012	0.017	0.000
20×10	2.556	0.007	0.046	0.013	0.008	0.000
20×20	1.727	0.000	0.019	0.010	0.010	0.000
50×5	3.351	0.450	0.374	0.028	0.021	0.128
50×10	3.572	0.228	0.224	0.072	0.105	-0.073
50×20	5.090	0.231	0.521	0.248	0.312	0.086
100×5	1.405	-0.218	-0.587	-0.776	-0.746	-0.775
100×10	1.764	-0.582	-0.652	-0.672	-0.657	-0.972
100×20	2.922	-0.336	-0.173	-0.119	-0.139	-0.741
200×10	0.452	-0.809	-1.021	-0.759	-0.763	-1.200
200×20	0.500	-1.245	-1.296	-0.917	-0.889	-1.591
500×20	-1.682	-2.594	-2.787	-2.438	-2.463	-2.835
平均值	2.005	-0.405	-0.438	-0.441	-0.432	-0.664

5.3. Performance of the proposed spatial dispersal model

5.3. 所提出空间分散模型的性能

The spatial dispersal is important for IWO, which determines the main searching ability of IWO. To show the performance of the proposed spatial dispersal, several computational experiments are designed as follows. Due to no other variants of IWO directly used to solve the BFSP, an effect discrete IWO (named SDIWO for short in this paper) proposed by Sang et al. (2018b) to solve the lot-streaming flow-shop scheduling problems is implemented in the first trail. The SDIWO algorithm adopts a spatial dispersal based on the distance of random insertion. To make a fair comparison, the same initial method with our proposed DIWO is used in SDIWO. In the second trial, our proposed DIWO algorithm is executed. Generally, the local search has important influence on the performance of the main algorithm. In order to purely demonstrate the searching ability of the proposed spatial dispersal, in the third and fourth trial, SDIWO without any local searches (named SDIWO_NL) and DIWO without any local searches (DIWO_NL) are executed, respectively. Through comparing the experimental results of these four trials, the contribution of the proposed spatial dispersal can be clearly ascertained. Moreover, Shao et al. (2017) proposed a self-adaptive discrete IWO (SaDIWO) to solve the BFSP with minimizing total tardiness, which has similar structure with our proposed DIWO. In the five trial, the spatial dispersal employed by SaDIWO is replaced with our proposed one, and a variant named SaDIWO_NS is developed. Meanwhile, we also adapt and implement SaDIWO to solve the considered BFSP. Based the comparison of SaDIWO_NS, SaDIWO, DIWO, we can further confirm the performance and applicability of our proposed spatial dispersal, and the worse performance of SaDIWO is whether attributed to its original spatial dispersal. The parameters of SDIWO and SaDIWO are consistent with its original paper. We run these compared algorithms for $\rho = 30$, that is $t = 30$ mn . Each instance is independently run 10 times. Computational results are given in Table 2.

空间分散对于 IWO 来说非常重要，它决定了 IWO 的主要搜索能力。为了展示所提出空间分散的性能，设计了以下几个计算实验。由于没有其他变体的 IWO 直接用于解决 BFSP，本文首次实现了 Sang 等人 (2018b) 提出的用于解决批量流车间调度问题的有效离散 IWO(在本文中简称为 SDIWO)。SDIWO 算法采用基于随机插入距离的空间分散。为了公平比较，SDIWO 使用了与本文提出的 DIWO 相同的初始方法。在第二次尝试中，执行了本文提出的 DIWO 算法。通常，局部搜索对主算法的性能有重要影响。为了纯粹展示所提出空间分散的搜索能力，在第三次和第四次尝试中，分别执行了没有局部搜索的 SDIWO(称为 SDIWO_NL) 和没有局部搜索的 DIWO(DIWO_NL)。通过比较这四次尝试的实验结果，可以清楚地确定所提出空间分散的贡献。此外，Shao 等人 (2017) 提出了一种用于解决 BFSP 并最小化总延迟的自适应离散 IWO(SaDIWO)，其结构与本文提出的 DIWO 相似。在第五次尝试中，将 SaDIWO 使用的空间分散替换为本文提出的分散，开发了一种名为 SaDIWO_NS 的变种。同时，我们还对 SaDIWO 进行了适应和实现，以解决考虑的 BFSP。基于 SaDIWO_NS、SaDIWO 和 DIWO 的比较，我们可以进一步确认所提出空间分散的性能和适用性，以及 SaDIWO 的性能较差是否归因于其原始的空间分散。SDIWO 和 SaDIWO 的参数与其原始论文保持一致。我们运行这些比较算法 $\rho = 30$，即 $t = 30$ mn 。每个实例独立运行 10 次。计算结果在表 2 中给出。

As can be seen from Table 2, DIWO achieves a smallest average ARPD of -0.664 . The SDIWO_NL yields the worst results among all compared algorithms. A 95% confidence interval plot is presented in Fig. 5 to show the performance of these algorithms. It can be clearly observed from this figure that DIWO outperforms other compared algorithms from a statistical point. Note that overlapping intervals denote statistically insignificant differences between the plotted overlapped means. So we can see from Fig. 5 that the DIWO_NL has the same performance with SDIWO since their confidence intervals almost coincide, whereas its results are worse than DIWO. It indicates that although DIWO_NL does not have local search to enhance its performance, it still has a very strong searching ability, which is because the proposed spatial dispersal stresses the balance between global exploration and local exploitation. In the proposed spatial dispersal, the better weeds in population have small values of d (i.e., the number of removed jobs), while the worse weeds have large values of d . The better weeds are used to generate small neighborhood to exploit the solution space, while the worse weeds are used to generate large neighborhood to explore the solution space. As also seen in Fig. 5, SDIWO_NL is much worse than SDIWO, which means that SDIWO_NL has poor searching ability. The performance of SDIWO fully depends on the local search. The spatial dispersal in SDIWO may only play a role of global exploration for solution space. Moreover, as can be seen in Table 4, SaDIWO is slightly inferior to SaDIWO_NS, which indicates that our proposed spatial dispersal can enhance the performance of SaDIWO but the enhancement is limited. It also reveals that the spatial dispersal is not the main reason for the worse performance of SaDIWO. Meanwhile, Fig. 5 shows that the differences between SaDIWO and SaDIWO_NL are not statistically significant, which further confirms the conclusions above. Therefore, the results above demonstrate that the proposed spatial dispersal has good performance and applicability, and can effectively enhance the performance of the algorithm.

如表 2 所示，DIWO 实现了最小的平均 ARP D 为-0.664。在所有比较的算法中，SDIWO_NL 的结果最差。图 5 展示了 95% 置信区间图，用以显示这些算法的性能。从图中可以明显看出，从统计学角度看，DIWO 优于其他比较的算法。需要注意的是，重叠的区间表示绘制的重叠均值之间的统计差异不显著。因此，我们可以从图 5 中看到 DIWO_NL 与 SDIWO 性能相同，因为它们的置信区间几乎重合，而其结果却比 DIWO 差。这表明尽管 DIWO_NL 没有局部搜索来提高其性能，但它仍然具有很强的搜索能力，这是因为所提出的空间分散强调了全局探索与局部开发之间的平衡。在所提出的空间分散中，种群中较好的草对应的 d (即被移除的工作数量) 值较小，而较差的草对应的 d 值较大。较好的草被用来生成小邻域来开发解空间，而较差的草被用来生成大邻域来探索解空间。如图 5 所示，SDIWO_NL 的性能比 SDIWO 差得多，这意味着 SDIWO_NL 的搜索能力较差。SDIWO 的性能完全依赖于局部搜索。SDIWO 中的空间分散可能仅在解空间的全局探索中起作用。此外，如表 4 所示，SaDIWO 略逊于 SaDIWO_NS，这表明我们提出的空间分散可以提高 SaDIWO 的性能，但提升有限。这也揭示了空间分散并非导致 SaDIWO 性能较差的主要原因。同时，图 5 显示 SaDIWO 与 SaDIWO_NL 之间的差异在统计学上不显著，这进一步证实了上述结论。因此，上述结果证明了所提出的空间分散具有良好的性能和适用性，能够有效提高算法的性能。

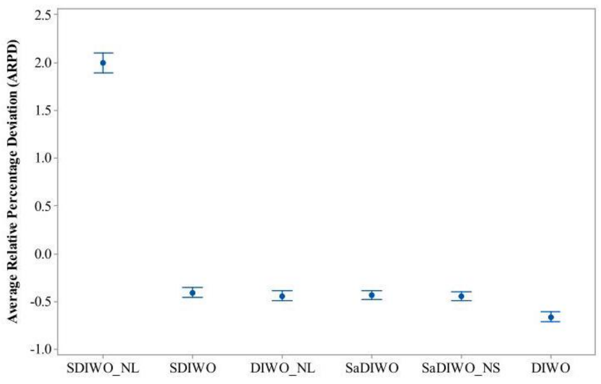

Fig. 5. Means plots and 95% confidence level intervals of SDIWO_NL, SDIWO, DIWO_NL, SaDIWO_NS, SaDIWO and DIWO.

图 5. SDIWO_NL、SDIWO、DIWO_NL、SaDIWO_NS、SaDIWO 和 DIWO 的均值图和 95% 置信水平区间。

5.4. Performance of the local search

5.4. 局部搜索的性能

In this section, we would like to evaluate the benefits of using the proposed local search and the speedup method. For this reason, some experiments are designed as follows. It should be noted that we regard the prune procedure as a part of the speedup method. The first trial is carried out to show the effectiveness of SRLS for the DIWO. The DIWO with SRLS (DIWO), the DIWO with RLS (DIWO_RLS) and the DIWO without any local searches (DIWO_NL), i.e., $pls = 0$, are executed. The second trial is carried out to test that the insertion neighborhood is more suitable than the swap neighborhood for BFSP. The insertion neighborhood in SRLS is replaced by the swap neighborhood. Since the speedup method in this paper is not suitable for evaluating the swap neighboring solutions, we adopt the speedup mechanism proposed by Li et al. (2009), which can save at least 50% computational time. This algorithm is denoted by DIWO_SP. The third trial is carried out to verify the effectiveness of the speedup method for SRLS. In this trial, SRLS without the speedup method (DIWO_NS) is executed. By comparing the results of these experiments, we can confirm the effectiveness of the SRLS and the speedup method. Additionally, SaDIWO mentioned in Section 5.3 also participates in comparison. Meanwhile, a variant of SaDIWO (denoted as SaDIWO_SRLS) is developed in the last trial, which replaces the local search (variable neighborhood search) of SaDIWO with SRLS. Through comparing SaDIWO_SRLS, SaDIWO and DIWO, we can further confirm the applicability and effectiveness of our proposed local search procedure, and the worse performance of SaDIWO is whether due to its own local search strategy. The termination criterion is also set $\rho = 30$, that is $t = 30$ mn . For each instance, all compared methods are run 10 times independently. The computational results are reported in Table 3.

在本节中，我们希望评估使用所提出的地方搜索和加速方法的益处。为此，设计了一些实验如下。需要注意的是，我们将剪枝过程视为加速方法的一部分。第一次尝试是为了展示 SRLS 对 DIWO 的有效性。执行了带有 SRLS 的 DIWO(DIWO)、带有 RLS 的 DIWO(DIWO_RLS) 以及没有地方搜索的 DIWO(DIWO_NL)，即 $pls = 0$ 。第二次尝试是为了测试插入邻域对于 BFSP 来说是否比交换邻域更合适。SRLS 中的插入邻域被替换为交换邻域。由于本文中的加速方法不适合评估交换邻域的解，我们采用了 Li 等人 (2009) 提出的加速机制，该机制至少可以节省 50% 计算时间。该算法被表示为 DIWO_SP。第三次尝试是为了验证加速方法对 SRLS 的有效性。在这个尝试中，执行了没有加速方法的 SRLS(DIWO_NS)。通过比较这些实验的结果，我们可以确认 SRLS 和加速方法的有效性。另外，第 5.3 节中提到的 SaDIWO 也参与了比较。同时，在最后一次尝试中，开发了一种 SaDIWO 的变体 (表示为 SaDIWO_SRLS)，它将 SaDIWO 的地方搜索 (变量邻域搜索) 替换为 SRLS。通过比较 SaDIWO_SRLS、SaDIWO 和 DIWO，我们可以进一步确认我们提出的地方搜索过程的适用性和有效性，以及 SaDIWO 的性能较差是否由于其自身的地方搜索策略。终止准则也设置为 $\rho = 30$ ，即 $t = 30$ mn 。对于每个实例，所有比较的方法都独立运行 10 次。计算结果报告在表 3 中。

As revealed in Table 3, the DIWO achieves the smallest ARPD with -0.664 compared to the corresponding values of -0.438, -0.142 , -0.050, -0.604, -0.540, -0.432 obtained by DIWO_NL, DIWO_SP, DIWO_NS, SaDIWO_SRLS, SaDIWO, and DIWO_RLS, respectively. It is obvious that the DIWO outperforms other compared algorithms. Fig. 6 displays the means plots and 95% confidence level intervals of all compared algorithms.

As seen in this figure, DIWO_NS performs very poor by comparing with other compared algorithms. This is because whether in the spatial dispersal or in the local search, a certain amount of partial sequences will be evaluated, which consumes considerable computational time so that all mechanisms of DIWO almost lose its efficacy. It can be also seen from Fig. 6 that DIWO_SP is inferior to DIWO_RLS and DIWO, which indicates that the insertion neighborhood is more effective than the swap neighborhood. Moreover, we also compare the effectiveness of RLS and SRLS for DIWO. As seen from Fig. 6, the confidence intervals of DIWO_RLS and DIWO do not completely overlap with each other, which indicates that SRLS is more suitable for DIWO than RLS. Meanwhile, we can clearly observe in Fig. 6 that DIWO is better than DIWO_NL, which reveals the proposed local search can effectively improve the performance of DIWO. Additionally, as seen from Table 3, SaDIWO is worse than DIWO and SaDIWO_SRLS, which indicates the proposed SRLS can enhance the performance of algorithm. Meanwhile, it also reveals that the worse performance of SaDIWO is attributed to the local search procedure. In the SaDIWO, an exhaustive variable neighborhood search is embedded, which consumes much time in one iteration, and largely influences the algorithm's ability to perform more iterations to explore more promising solutions. Our proposed SRLS is a lightweight local search mechanism, which cannot only ensure sufficient exploitation for local area, but also does not influence the exploration ability of algorithm. It can be also observed from Fig. 6 that DIWO and SaDIWO_NLS are significantly better than SaDIWO, which further confirms the conclusions above. Therefore, on the basis of the results and analyses above, it can be concluded that SRLS has good performance and applicability, and can effectively enhance the performance of the proposed algorithm.

如表 3 所示,DIWO 实现了最小的 ARPD,为-0.664,而 DIWO_NL、DIWO_SP、DIWO_NS、SaDIWO_SRLS、SaDIWO 和 DIWO_RLS 分别对应的值为-0.438、-0.142、$-0.050, -0.604, -0.540, -0.432$ 。显然，DIWO 优于其他对比算法。图 6 展示了所有对比算法的平均值图和 95% 置信水平区间。从图中可以看出, DIWO_NS 与其他对比算法相比表现很差。这是因为无论是在空间分散还是在局部搜索中，都需要评估一定数量的部分序列，这消耗了大量的计算时间，导致 DIWO 的所有机制几乎失效。从图 6 还可以看出，DIWO_SP 不如 DIWO_RLS 和 DIWO，这表明插入邻域比交换邻域更有效。此外，我们还比较了 DIWO 中 RLS 和 SRLS 的有效性。从图 6 可以看出，DIWO_RLS 和 DIWO 的置信区间并不完全重叠，这表明 SRLS 比 RLS 更适合 DIWO。同时，我们可以在图 6 中清楚地看到 DIWO 优于 DIWO_NL，这揭示了提出的局部搜索可以有效地提高 DIWO 的性能。另外，从表 3 可以看出，SaDIWO 的性能低于 DIWO 和 SaDIWO_SRLS，这表明提出的 SRLS 可以增强算法的性能。同时，这也揭示了 SaDIWO 性能较差的原因在于局部搜索过程。在 SaDIWO 中，嵌入了一个详尽的变量邻域搜索，这在一个迭代中消耗了大量时间，并大大影响了算法进行更多迭代以探索更有希望解的能力。我们提出的 SRLS 是一种轻量级的局部搜索机制，不仅能确保对局部区域进行充分的开发，而且不会影响算法的探索能力。从图 6 还可以看出，DIWO 和 SaDIWO_NLS 明显优于 SaDIWO，这进一步证实了上述结论。因此，基于上述结果和分析，可以得出结论，SRLS 具有良好的性能和适用性，能够有效提高所提出算法的性能。

Table 3
表 3
The ARPD of DIWO_NL, DIWO_SP, DIWO_NS, DIWO_RLS, SaDIWO_SRLS, SaDIWO and DIWO.
DIWO_NL、DIWO_SP、DIWO_NS、DIWO_RLS、SaDIWO_SRLS、SaDIWO 和 DIWO 的 ARPD。

Instance	DIWO_NL	DIWO_SP	DIWO_NS	DIWO_RLS	SaDIWO_SRLS	SaDIWO	DIWO
20×5	0.078	0.068	0.049	0.045	0.038	0.017	0.000
20×10	0.046	0.049	0.003	0.002	0.015	0.008	0.000
20×20	0.019	0.018	0.001	0.000	0.005	0.010	0.000
50×5	0.374	0.303	0.566	0.152	0.305	0.021	0.128
50×10	0.224	0.276	0.352	0.021	0.100	0.105	-0.073
50×20	0.521	0.683	0.571	0.162	0.239	0.312	0.086
100×5	-0.587	-0.440	0.171	-0.709	-0.622	-0.746	-0.775
100×10	-0.652	-0.304	0.079	-0.916	-0.840	-0.657	-0.972
100×20	-0.173	0.463	0.616	-0.612	-0.589	-0.139	-0.741
200×10	-1.021	-0.469	-0.262	-1.143	-0.975	-0.763	-1.200
200×20	-1.296	-0.284	-0.431	-1.477	-1.464	-0.889	-1.591
500×20	-2.787	-2.070	-2.320	-2.777	-2.698	-2.463	-2.835
Average	-0.438	-0.142	-0.050	-0.604	-0.540	-0.432	-0.664

实例	DIWO_NL	DIWO_SP	DIWO_NS	DIWO_RLS	SaDIWO_SRLS	SaDIWO	DIWO
20×5	0.078	0.068	0.049	0.045	0.038	0.017	0.000
20×10	0.046	0.049	0.003	0.002	0.015	0.008	0.000
20×20	0.019	0.018	0.001	0.000	0.005	0.010	0.000
50×5	0.374	0.303	0.566	0.152	0.305	0.021	0.128
50×10	0.224	0.276	0.352	0.021	0.100	0.105	-0.073
50×20	0.521	0.683	0.571	0.162	0.239	0.312	0.086
100×5	-0.587	-0.440	0.171	-0.709	-0.622	-0.746	-0.775
100×10	-0.652	-0.304	0.079	-0.916	-0.840	-0.657	-0.972
100×20	-0.173	0.463	0.616	-0.612	-0.589	-0.139	-0.741
200×10	-1.021	-0.469	-0.262	-1.143	-0.975	-0.763	-1.200
200×20	-1.296	-0.284	-0.431	-1.477	-1.464	-0.889	-1.591
500×20	-2.787	-2.070	-2.320	-2.777	-2.698	-2.463	-2.835
平均值	-0.438	-0.142	-0.050	-0.604	-0.540	-0.432	-0.664

5.5. Comparison of other algorithms

5.5. 其他算法的比较

To investigate the effectiveness of the proposed DIWO, we compare it with several state-of-the-art algorithms published in recent literature. These algorithms include HDDE (Wang et al., 2010a), hmgHS (Wang et al., 2011), IG (Ribas et al., 2011), TPA (Wang et al., 2012), RAIS (Lin and Ying, 2013), MA (Pan et al., 2013b), SVNS_D (Ribas et al., 2013), HVNS (Moslehi and Khorasanian, 2014a), DE-ABC (Han et al., 2015b), DE_PLS (Tasgetiren et al., 2015), MFFO (Han et al., 2016a), SaDIWO (Shao et al., 2017), and P-EDA (Shao et al., 2018a). As mentioned from Črepinšek et al. (2014), all experiments should be carried out and compared under the same or stricter conditions. If different algorithms are implemented on different computers, the implementation settings of different algorithms will be different. The running time of each algorithm may be also affected by the operating system and other applications running the experiments. Additionally, these algorithms were often coded by different programming languages and adopted different termination conditions. The reported results in the literature come from different upper bounds. Thus, to make a fair and reasonable comparison, we strictly re-implement all compared algorithms with same programming language and execute them on the same computer environment. We can ensure that all algorithms have same CPU power and time available. All algorithms adopt the same termination criterion $t = \rho$mn , where ρ is tested at three values 30,60 and 90 to test how the different algorithms perform with different CPU time. This termination criterion has been increasingly used in recent literature on scheduling (Karabulut, 2016; Pan et al., 2013b; Ribas et al., 2015). Each instance is independently executed 10 replications. It should be noted that TPA is terminated when reaching the maximum number of iterations in its original paper. The performance of TPA largely depends on the maximum number of iterations, so we set it as 1.8×10^6 based on the termination criterion considered. As suggested in Ribas et al. (2013), the parameters of SVNS_D are taken as $\alpha = 0.75, \beta = 0.5$ and $ds = 8$. We also use the PF-NEH(x)heuristic to generate the initial solution for SVNS_D. The values of parameters used for other re-implemented algorithms are recommended by their respective original works. Moreover, SaDIWO is initially used to solve BFSP with total tardiness, so we adapt its objective evolution function, and then use the same initialization strategy with the proposed DIWO to generate its initial population. Meanwhile, the adopted speedup method is also incorporated into it. Tables 4 to 6 summarize the ARPD of all compared algorithms for each termination criterion. Best values are printed in boldface in these tables.

为了调查所提出 DIWO 的有效性，我们将其与近期文献中发表的几种最先进的算法进行了比较。这些算法包括 HDDE(Wang et al., 2010a)、hmgHS(Wang et al., 2011)、IG(Ribas et al., 2011)、TPA(Wang et al., 2012)、RAIS(Lin 和 Ying, 2013)、MA(Pan et al., 2013b)、SVNS_D(Ribas et al., 2013)、HVNS(Moslehi 和 Khorasanian, 2014a)、DE-ABC(Han et al., 2015b)、DE_PLS(Tasgetiren et al., 2015)、MFFO(Han et al., 2016a)、SaDIWO(Shao et al., 2017) 和 P-EDA(Shao et al., 2018a)。如 Črepinšek et al.(2014) 所述，所有实验都应在相同或更严格的条件下进行并比较。如果不同的算法在不同的计算机上实现，那么不同算法的实现设置将有所不同。每种算法的运行时间可能还会受到操作系统和其他运行实验的应用程序的影响。此外，这些算法通常由不同的编程语言编写并采用了不同的终止条件。文献中报告的结果来自于不同的上限。因此，为了进行公平合理的比较，我们严格地用相同的编程语言重新实现了所有比较的算法，并在相同的计算机环境下执行它们。我们可以确保所有算法拥有相同的 CPU 性能和时间。所有算法都采用相同的终止标准 $t = \rho$mn ，其中 ρ 在 30、60 和 90 三个值上进行测试，以检验不同算法在不同 CPU 时间下的表现。这个终止标准在近期关于调度的文献中 (Karabulut, 2016; Pan et al., 2013b; Ribas et al., 2015) 被越来越频繁地使用。每个实例独立执行 10 次复制。需要注意的是，TPA 在其原始论文中达到最大迭代次数时终止。TPA 的性能在很大程度上取决于最大迭代次数，因此我们根据考虑的终止标准将其设置为 1.8×10^6 。如 Ribas et al.(2013) 建

议，SVNS_D 的参数取值为 $\alpha = 0.75, \beta = 0.5$ 和 $ds = 8$ 。我们还使用 PF-NEH(x) 启发式方法为 SVNS_D 生成初始解。其他重新实现算法所使用的参数值是由它们各自原始作品推荐的。此外，SaDIWO 最初用于解决总迟到的 BFSP 问题，因此我们调整了其目标进化函数，并使用与所提出 DIWO 相同的初始化策略来生成其初始种群。同时，采用的速度提升方法也融入其中。表 4 至表 6 总结了所有比较算法在每个终止标准下的 ARPD。表中的最佳值以粗体字打印。

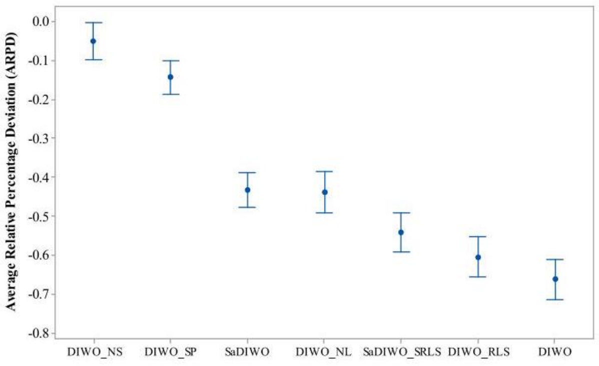

Fig. 6. Means plots and 95% confidence level intervals of DIWO_NL, DIWO_SP, DIWO_NS, DIWO_RLS, SaDIWO_SRLS, SaDIWO and DIWO algorithm.

图 6. DIWO_NL、DIWO_SP、DIWO_NS、DIWO_RLS、SaDIWO_SRLS、SaDIWO 以及 DIWO 算法的均值图和 95% 置信水平区间。

It is clear from Tables 4 to 6 that the presented DIWO achieves the lowest overall average ARPD for all elapsed CPU time. MA, DE_PLS and SaDIWO are also competitive for some instances. To be specific, for $\rho = 30$ termination criterion (Table 4), only DIWO could always obtain the best solutions on small instances (i.e., 20 jobs). MA has good performance on 50×5 and 50×20 . For large instances (i.e.,200 and 500 jobs), MA, SaDIWO, P-EDA and DIWO are more competitive than other algorithms. For $\rho = 60$ termination criterion, it can be seen from Table 5 that DIWO has good performance on most instances, except for $50 \times 5, 50 \times 20, 100 \times 5$ and 100×20 . For these instances, MA, TPA and SaDIWO yield the good results, respectively. For the small instances (i.e., 20 jobs), P-EDA and DIWO achieves the best results. For the last termination criterion $\rho = 90$, DIWO has good performance on small and large instances, but for instances that have 50 jobs, DIWO is slightly inferior to MA and DE_PLS. Additionally, taken together from all termination criteria, for two variants of variable neighborhood search, HVNS is better than SVNS_D. As a swarm intelligence optimization algorithm, MFFO provides better results than HDDE, hmgHS and IG. For the non-population based algorithms, i.e., TPA, SVNS_D, HVNS, DE_PLS, and IG, TPA obtains the best performance. For another discrete variant of IWO, i.e., SaDIWO, it has similar framework with DIWO, but its reproduction, spatial dispersal and local intensification phases are different from DIWO. Especially for local intensification phase, SaDIWO adopts a variable neighborhood search (VNS) based on insertion, swap, and double-edge insertion. From the results in Tables 4 to 6, SaDIWO is inferior to DIWO. The main reason behind this is that the VNS of SaDIWO consume much time, which diminishes the algorithm's capability to perform more iterations so that the exploration time suffers a great compression. That is to say, over local intensification influences the balance between exploration and exploitation. Another main reason is that the type of the neighborhood structure also influences the performance of algorithm. In Section 5.4, it has been validated that the swap neighborhood structure could not play a decisive role for BFSP with makespan. The swap neighborhood structure consumes a certain search time for SaDIWO. Instead, DIWO employs a single insertion neighborhood structure and controls it with probability way, which effectively implements local intensification and ensures the balance between exploitation and exploration. Additionally, as an evolutionary algorithm based on statistics, P-EDA is also very competitive, but which is still inferior to DIWO. Compared with DIWO, P-EDA is relatively complicated, and needs to construct a probabilistic model for a promising population. The accuracy of the probabilistic model largely influences the performance of P-EDA. DIWO explores the whole solution space only through a simple random insertion mechanism, which makes DIWO has lower complexity. Meanwhile, an effective local search ensures the local exploration ability of DIWO.

从表 4 到表 6 可以明显看出，所提出的 DIWO 在所有已过的 CPU 时间中实现了最低的整体平均 ARPDU。MA、DE_PLS 和 SaDIWO 在某些实例上也有竞争力。具体来说，对于 $\rho = 30$ 终止准则 (表 4)，只有 DIWO 始终能在小实例 (即 20 个作业) 上获得最佳解。MA 在 50×5 和 50×20 上表现良好。对于大实例 (即 200 和 500 个作业)，MA、SaDIWO、P-EDA 和 DIWO 比其他算法更有竞争力。对于 $\rho = 60$ 终止准则，从表 5 可以看出，DIWO 在大多数实例上表现良好，除了 $50 \times 5, 50 \times 20, 100 \times 5$ 和 100×20 。对于这些实例，MA、TPA 和 SaDIWO 分别获得了良好的结果。对于小实例 (即 20 个作业)，P-EDA 和 DIWO

取得了最佳结果。对于最后一个终止准则 $\rho = 90$ ，DIWO 在小实例和大实例上都有良好表现，但对于有 50 个作业的实例，DIWO 略逊于 MA 和 DE_PLS。此外，从所有终止准则综合来看，对于变量邻域搜索的两个变种，HVNS 优于 SVNS_D。作为一种群体智能优化算法，MFFO 提供了比 HDDE、hmgHS 和 IG 更好的结果。对于基于非种群的算法，即 TPA、SVNS_D、HVNS、DE_PLS 和 IG，TPA 获得了最佳性能。对于 IWO 的另一个离散变种，即 SaDIWO，它与 DIWO 具有相似的框架，但其繁殖、空间分散和局部强化阶段与 DIWO 不同。特别是在局部强化阶段，SaDIWO 采用基于插入、交换和双边缘插入的变量邻域搜索 (VNS)。从表 4 到表 6 的结果来看，SaDIWO 不如 DIWO。这背后的主要原因在于，SaDIWO 的 VNS 消耗了大量时间，这减少了算法执行更多迭代的 capability，从而导致探索时间受到很大的压缩。也就是说，过度局部强化影响了探索和利用之间的平衡。另一个主要原因是邻域结构类型也影响了算法的性能。在第 5.4 节中，已经验证了交换邻域结构对于具有最短完工时间的 BFSP 不起决定性作用。交换邻域结构为 SaDIWO 消耗了一定的搜索时间。相反，DIWO 采用单一插入邻域结构，并通过概率方式控制它，有效地实现了局部强化并确保了利用和探索之间的平衡。此外，作为一种基于统计的进化算法，P-EDA 也非常有竞争力，但仍然不如 DIWO。与 DIWO 相比，P-EDA 相对复杂，需要为有希望的种群构建一个概率模型。概率模型的准确性很大程度上影响了 P-EDA 的性能。DIWO 仅通过简单的随机插入机制探索整个解空间，这使得 DIWO 具有较低的复杂性。同时，有效的局部搜索确保了 DIWO 的局部探索能力。

Table 4
表 4
The ARPD of the all compared algorithms $(\rho = 30)$.
所有比较算法的平均相对误差百分比 (ARPD) $(\rho = 30)$ 。

Instance	HDDE	hmgHS	IG	TPA	RAIS	MA	SVNS_D	HVNS	DE-ABC	DE_PLS	MFFO	SaDIWO	P-EDA	DIWO
20 × 5	0.013	0.121	0.065	0.008	0.230	0.005	0.069	0.025	0.014	0.007	0.005	0.017	0.060	0.000
20 × 10	0.012	0.057	0.107	0.014	0.079	0.007	0.034	0.004	0.016	0.006	0.007	0.008	0.009	0.000
20 × 20	0.001	0.031	0.104	0.000	0.017	0.001	0.003	0.000	0.009	0.001	0.003	0.010	0.000	0.000
50 × 5	0.414	0.574	1.220	0.346	0.634	0.058	0.196	0.298	1.842	0.229	0.705	0.021	0.155	0.128
50 × 10	0.191	0.398	1.374	-0.045	0.336	-0.039	0.296	0.086	1.463	-0.040	0.395	0.105	0.040	-0.073
50 × 20	0.223	0.360	1.429	-0.063	0.978	0.009	0.561	0.037	1.210	-0.016	0.342	0.312	0.104	0.086
100 × 5	1.354	0.719	0.929	-0.284	0.062	-0.405	-0.609	-0.351	2.428	0.111	0.916	-0.746	-0.647	-0.775
100 × 10	0.596	0.127	1.041	-0.830	0.095	-0.811	-0.465	-0.696	1.568	-0.357	0.135	-0.657	-0.800	-0.972
100 × 20	0.403	0.090	1.149	-0.951	1.765	-0.780	0.120	-0.734	1.211	-0.387	0.016	-0.139	-0.616	-0.741
200 × 10	2.059	1,727	1.935	-0.789	1.722	-0.737	-0.742	-0.530	2.286	-0.224	1.180	-0.763	-0.939	-1.200
200 × 20	0.909	0.614	1.282	-1.480	2.802	-1.343	-0.857	-1.191	1.045	-0.606	-0.028	-0.889	-0.281	-1.591
500 × 20	1.171	1.078	1.376	-1.933	4.083	-2.540	-2.425	-0.148	0.768	-2.139	0.038	-2.463	-2.583	-2.835
Average	0.612	0.491	1.001	-0.501	1.067	-0.548	-0.318	-0.267	1.155	-0.285	0.310	-0.432	-0.542	-0.664

实例	HDDE	hmgHS	IG	TPA	RAIS	MA	SVNS_D	HVNS	DE-ABC	DE_PLS	MFFO	SaDIWO 由于提供的文本是缩写或代码，没有具体的学术文本内容，因此无法进行翻译。上述输出仅保留了原文的缩写或代码，并按照要求删除了 text 字段。如果这些缩写代表特定的学术概念或术语，需要具体的全称或上下文才能进行准确翻译。	P-EDA	DIWO
20 × 5	0.013	0.121	0.065	0.008	0.230	0.005	0.069	0.025	0.014	0.007	0.005	0.017	0.060	0.000
20 × 10	0.012	0.057	0.107	0.014	0.079	0.007	0.034	0.004	0.016	0.006	0.007	0.008	0.009	0.000
20 × 20	0.001	0.031	0.104	0.000	0.017	0.001	0.003	0.000	0.009	0.001	0.003	0.010	0.000	0.000
50 × 5	0.414	0.574	1.220	0.346	0.634	0.058	0.196	0.298	1.842	0.229	0.705	0.021	0.155	0.128
50 × 10	0.191	0.398	1.374	-0.045	0.336	-0.039	0.296	0.086	1.463	-0.040	0.395	0.105	0.040	-0.073
50 × 20	0.223	0.360	1.429	-0.063	0.978	0.009	0.561	0.037	1.210	-0.016	0.342	0.312	0.104	0.086
100 × 5	1.354	0.719	0.929	-0.284	0.062	-0.405	-0.609	-0.351	2.428	0.111	0.916	-0.746	-0.647	-0.775
100 × 10	0.596	0.127	1.041	-0.830	0.095	-0.811	-0.465	-0.696	1.568	-0.357	0.135	-0.657	-0.800	-0.972
100 × 20	0.403	0.090	1.149	-0.951	1.765	-0.780	0.120	-0.734	1.211	-0.387	0.016	-0.139	-0.616	-0.741
200 × 10	2.059	1,727	1.935	-0.789	1.722	-0.737	-0.742	-0.530	2.286	-0.224	1.180	-0.763	-0.939	-1.200
200 × 20	0.909	0.614	1.282	-1.480	2.802	-1.343	-0.857	-1.191	1.045	-0.606	-0.028	-0.889	-0.281	-1.591
500 × 20	1.171	1.078	1.376	-1.933	4.083	-2.540	-2.425	-0.148	0.768	-2.139	0.038	-2.463	-2.583	-2.835
平均值	0.612	0.491	1.001	-0.501	1.067	-0.548	-0.318	-0.267	1.155	-0.285	0.310	-0.432	-0.542	-0.664

Table 5
表 5
The ARPD of the all compared algorithms $(\rho = 60)$.
所有比较算法的平均相对误差百分比 (ARPD) $(\rho = 60)$ 。

Instance	HDDE	hmgHS	IG	TPA	RAIS	MA	SVNS_D	HVNS	DE-ABC	DE_PLS	MFFO	SaDIWO	P_EDA	DIWO
20 × 5	0.004	0.094	0.031	0.012	0.227	0.000	0.050	0.017	0.007	0.000	0.002	0.012	0.000	0.000
20 × 10	0.008	0.039	0.068	0.028	0.080	0.005	0.007	0.002	0.012	0.006	0.003	0.003	0.000	0.000
20 × 20	0.002	0.014	0.058	0.000	0.017	0.000	0.008	0.000	0.004	0.000	0.000	0.004	0.000	0.000
50 × 5	0.237	0.438	1.085	0.317	0.575	-0.074	0.147	0.202	1.695	0.062	0.563	-0.066	0.050	0.016
50 × 10	0.069	0.347	1.200	-0.028	0.283	-0.081	0.120	-0.036	1.376	-0.113	0.284	-0.004	-0.037	-0.115
50 × 20	0.078	0.312	1.293	-0.018	0.363	-0.051	0.497	-0.053	1.115	-0.130	0.232	0.212	0.136	0.000
100 × 5	0.985	0.401	0.699	-0.324	0.048	-0.577	-0.733	-0.554	2.277	-0.210	0.682	-0.926	-0.813	-0.809
100 × 10	0.270	-0.086	0.814	-0.831	-0.178	-1.002	-0.638	-0.834	1.495	-0.618	-0.025	-0.868	-0.951	-1.060
100 × 20	0.127	-0.161	0.924	-0.966	0.036	-0.848	-0.134	-0.911	1.103	-0.636	-0.161	-0.396	-0.695	-0.848
200 × 10	1.816	1.446	1.542	-0.786	0.587	-0.891	-0.850	-0.732	2.190	-0.347	0.810	-0.919	-1.070	-1.285
200 × 20	0.645	0.437	0.916	-1.555	0.419	-1.482	-0.986	-1.419	0.961	-0.833	-0.262	-1.118	-1.442	-1.691
500 × 20	0.957	0.933	1.270	-2.030	1.454	-2.623	-2.483	-1.883	0.595	-2.230	-0.243	-2.464	-2.669	-2.925
Average	0.433	0.351	0.825	-0.515	0.326	-0.635	-0.416	-0.517	1.069	-0.421	0.157	-0.544	-0.624	-0.726

实例	HDDE	hmgHS	IG	TPA	RAIS	MA	SVNS_D	HVNS	DE-ABC	DE_PLS	MFFO	SaDIWO	P_EDA	DIWO
20 × 5	0.004	0.094	0.031	0.012	0.227	0.000	0.050	0.017	0.007	0.000	0.002	0.012	0.000	0.000
20 × 10	0.008	0.039	0.068	0.028	0.080	0.005	0.007	0.002	0.012	0.006	0.003	0.003	0.000	0.000
20 × 20	0.002	0.014	0.058	0.000	0.017	0.000	0.008	0.000	0.004	0.000	0.000	0.004	0.000	0.000
50 × 5	0.237	0.438	1.085	0.317	0.575	-0.074	0.147	0.202	1.695	0.062	0.563	-0.066	0.050	0.016
50 × 10	0.069	0.347	1.200	-0.028	0.283	-0.081	0.120	-0.036	1.376	-0.113	0.284	-0.004	-0.037	-0.115
50 × 20	0.078	0.312	1.293	-0.018	0.363	-0.051	0.497	-0.053	1.115	-0.130	0.232	0.212	0.136	0.000
100 × 5	0.985	0.401	0.699	-0.324	0.048	-0.577	-0.733	-0.554	2.277	-0.210	0.682	-0.926	-0.813	-0.809
100 × 10	0.270	-0.086	0.814	-0.831	-0.178	-1.002	-0.638	-0.834	1.495	-0.618	-0.025	-0.868	-0.951	-1.060
100 × 20	0.127	-0.161	0.924	-0.966	0.036	-0.848	-0.134	-0.911	1.103	-0.636	-0.161	-0.396	-0.695	-0.848
200 × 10	1.816	1.446	1.542	-0.786	0.587	-0.891	-0.850	-0.732	2.190	-0.347	0.810	-0.919	-1.070	-1.285
200 × 20	0.645	0.437	0.916	-1.555	0.419	-1.482	-0.986	-1.419	0.961	-0.833	-0.262	-1.118	-1.442	-1.691
500 × 20	0.957	0.933	1.270	-2.030	1.454	-2.623	-2.483	-1.883	0.595	-2.230	-0.243	-2.464	-2.669	-2.925
平均值	0.433	0.351	0.825	-0.515	0.326	-0.635	-0.416	-0.517	1.069	-0.421	0.157	-0.544	-0.624	-0.726

Table 6
表 6
The ARPD of the all compared algorithms ($\rho = 90$) .
所有比较算法的平均相对误差百分比 (ARPD) ($\rho = 90$) 。

Instance	HDDE	hmgHS	IG	TPA	RAIS	MA	SVNS_D	HVNS	DE-ABC	DE_PLS	MFFO	SaDIWO	P-EDA	DIWO
20 × 5	0.001	0.077	0.010	0.013	0.246	0.000	0.036	0.013	0.001	0.003	0.000	0.000	0.000	0.000
20 × 10	0.008	0.033	0.050	0.027	0.064	0.003	0.003	0.000	0.008	0.002	0.002	0.000	0.000	0.000
20 × 20	0.000	0.010	0.033	0.000	0.020	0.000	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.000
50 × 5	0.148	0.365	0.998	0.292	0.556	-0.111	0.097	0.149	1.644	-0.062	0.447	-0.168	0.054	-0.080
50 × 10	-0.049	0.265	1.083	-0.036	0.291	-0.141	0.126	-0.067	1.277	-0.255	0.218	-0.057	-0.067	-0.173
50 × 20	-0.004	0.275	1.209	-0.032	0.394	-0.034	0.441	-0.073	1.078	-0.210	0.222	0.191	0.092	-0.094
100 × 5	0.737	0.211	0.591	-0.340	0.019	-0.836	-0.817	-0.553	2.188	-0.281	0.393	-1.053	-0.858	-0.948
100 × 10	0.019	-0.191	0.616	-0.903	-0.243	-1.051	-0.758	-0.879	1.436	-0.823	-0.173	-0.938	-0.107	-1.068
100 × 20	-0.024	-0.222	0.783	-0.963	-0.109	-0.958	-0.247	-0.965	1.064	-0.819	-0.230	-0.555	-0.767	-0.971
200 × 10	1.670	1,324	1.298	-0.810	0.473	-0.988	-0.904	-0.743	2.121	-0.414	0.630	-1.038	-1.169	-1.359
200 × 20	0.532	0.266	0.721	-1.579	0.123	-1.580	-1.110	-1.468	0.934	-0.944	-0.407	-1.262	-1.543	-1.782
500 × 20	0.874	0.863	1.041	-2.053	0.936	-2.676	-2.529	-1.976	0.480	-2.245	-0.414	-2.547	-2.710	-2.970
Average	0.326	0.273	0.703	-0.532	0.231	-0.698	-0.472	-0.547	1.019	-0.504	0.057	-0.617	-0.633	-0.787

实例	高维数据驱动嵌入 (HDDE)	人类微生物组项目相关的高通量测序 (hmgHS)	免疫球蛋白 (IG)	组织型纤溶酶原激活剂 (TPA)	随机访问信息系统 (RAIS)	多元分析 (MA)	支持向量机噪声分解 (SVNS_D)	混合变量邻域搜索 (HVNS)	差分进化-人工蜂群算法 (DE-ABC)	差分进化-偏最小二乘法 (DE_PLS)	MFFO	SaDIWO	P-EDA	DIWO
20 × 5	0.001	0.077	0.010	0.013	0.246	0.000	0.036	0.013	0.001	0.003	0.000	0.000	0.000	0.000
20 × 10	0.008	0.033	0.050	0.027	0.064	0.003	0.003	0.000	0.008	0.002	0.002	0.000	0.000	0.000
20 × 20	0.000	0.010	0.033	0.000	0.020	0.000	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.000
50 × 5	0.148	0.365	0.998	0.292	0.556	-0.111	0.097	0.149	1.644	-0.062	0.447	-0.168	0.054	-0.080
50 × 10	-0.049	0.265	1.083	-0.036	0.291	-0.141	0.126	-0.067	1.277	-0.255	0.218	-0.057	-0.067	-0.173
50 × 20	-0.004	0.275	1.209	-0.032	0.394	-0.034	0.441	-0.073	1.078	-0.210	0.222	0.191	0.092	-0.094
100 × 5	0.737	0.211	0.591	-0.340	0.019	-0.836	-0.817	-0.553	2.188	-0.281	0.393	-1.053	-0.858	-0.948
100 × 10	0.019	-0.191	0.616	-0.903	-0.243	-1.051	-0.758	-0.879	1.436	-0.823	-0.173	-0.938	-0.107	-1.068
100 × 20	-0.024	-0.222	0.783	-0.963	-0.109	-0.958	-0.247	-0.965	1.064	-0.819	-0.230	-0.555	-0.767	-0.971
200 × 10	1.670	1,324	1.298	-0.810	0.473	-0.988	-0.904	-0.743	2.121	-0.414	0.630	-1.038	-1.169	-1.359
200 × 20	0.532	0.266	0.721	-1.579	0.123	-1.580	-1.110	-1.468	0.934	-0.944	-0.407	-1.262	-1.543	-1.782
500 × 20	0.874	0.863	1.041	-2.053	0.936	-2.676	-2.529	-1.976	0.480	-2.245	-0.414	-2.547	-2.710	-2.970
平均值	0.326	0.273	0.703	-0.532	0.231	-0.698	-0.472	-0.547	1.019	-0.504	0.057	-0.617	-0.633	-0.787

Additionally, from the results above, we can see that the compared algorithms have different performance on different scale instances and termination criterion. To investigate these behaviors, Fig. 7 firstly shows the interaction plot between algorithm and the number of jobs(n) with machines(m). It can be seen from this figure that all compared algorithms are more influenced by m and n . In more detail, most algorithms perform better when n increases, except for RAIS, hmgHS , HDDE and IG. For the interaction of algorithm and m , we can observe that IG and RAIS perform worse and worse with the increase of the number of machines, while other algorithms yield the opposite results. Meanwhile, we can also see that the proposed DIWO performs better and better with the increase of n and m , which also reveals DIWO is good at solving the large instances. Moreover, to investigate the behavior of the algorithms on different termination criteria, the interaction between algorithm and ρ is shown in Fig. 8. As seen this figure, the performance of RAIS is rather poor for $\rho = 30$, but it achieves good results for $\rho = 60$ and $\rho = 90$. The RAIS is more influenced by ρ and the reason for this may lie in that no effective local search in it so that it requires more time to get better results. The TPA obtains the similar results for each value of ρ , which means that TPA converges rapidly and substantial additional CPU time is not helpful to enhance its performance. For other algorithms, additional running time can help them obtain better results, but the improvement is very limited. In other words, these algorithms cannot be expect to behave entirely different through adding running time. Their searching mechanism largely determines their performance.

此外，从上述结果中，我们可以看到比较的算法在不同规模的实例和终止准则上表现不同。为了研究这些行为，图 7 首先展示了算法与作业数量 (n) 及机器数量 (m) 之间的交互图。从图中可以看出，所有比较的算法都受到 m 和 n 的影响。更详细地说，当 n 增加时，大多数算法表现更好，RAIS、hmgHS 、HDDE 和 IG 除外。对于算法与 m 的交互，我们可以观察到 IG 和 RAIS 随着机器数量的增加表现越来越差，而其他算法则呈现出相反的结果。同时，我们还可以看到提出的 DIWO 随着, n 和 m 的增加表现越来越好，这也揭示了 DIWO 擅长解决大规模实例。此外，为了研究算法在不同终止准则下的行为，图 8 展示了算法与 ρ 之间的交互。从图中可以看到，RAIS 在 $\rho = 30$ 下的表现相当差，但在 $\rho = 60$ 和 $\rho = 90$ 下取得了良好的结果。RAIS 受 ρ 的影响更大，这可能是因为其中没有有效的局部搜索，因此需要更多时间来获得更好的结果。TPA 对每个 ρ 的值都获得了类似的结果，这意味着 TPA 收敛迅速，额外的 CPU 时间对其性能提升不大。对于其他算法，额外的运行时间可以帮助它们获得更好的结果，但改进非常有限。

换句话说，这些算法通过增加运行时间不太可能有完全不同的表现。它们的搜索机制在很大程度上决定了它们的性能。

Although the results in Tables 4 to 6 have demonstrated the effectiveness of the proposed DIWO, due to the stochastic characteristic of the compared algorithms, it is essential to carry out a comprehensive statistical experiment to confirm whether the observed differences among the compared algorithms are statistically meaningful. In the statistical experiment, three types of statistical test are carried out, which include two parametric tests and a non-parametric test. Firstly, the parametric ANOVA test is used to check whether the ARPD obtained by all compared algorithms have statistically significant differences, where the heuristic type (algorithm) is considered as a factor. Therefore, we carry out three ANOVA tests in this experiment. All tests are run with a confidence level of 95% $(\alpha = 0.05)$. The results of ANOVA are reported by using the form of ANOVA charts with 95% confidence intervals, which are shown in Fig. 9. It is remarkable that overlapping intervals of two algorithms indicate that there is no significant difference between their performance. It can be seen from Fig. 9 that there are no overlaps between DIWO and other compared algorithms for each termination criterion, which means the differences between DIWO and other algorithms are statistically significant at the 95% confidence level. Thus, we can conclude that DIWO significantly outperforms other compared methods for all termination criteria. Additionally, for $\rho = 60$ and $\rho = 90$, the confidence intervals of MA and P-EDA are almost completely overlapped, which reveals that they have similar performance on these termination criteria. TPA, HVNS, DE_PLS and SVNS_D also have overlapped confidence intervals when $\rho = 90$.

尽管表 4 至表 6 的结果已经证明了所提出 DIWO 的有效性，但由于比较算法的随机特性，进行一项全面的统计实验来确认比较算法之间的观察差异是否具有统计学意义是必要的。在统计实验中，进行了三种类型的统计测试，包括两种参数测试和一个非参数测试。首先，参数方差分析 (ANOVA) 测试被用来检查所有比较算法获得的 ARPD 是否存在统计上的显著差异，其中启发式类型 (算法) 被视为一个因素。因此，我们在实验中进行了三次 ANOVA 测试。所有测试都以 95% $(\alpha = 0.05)$ 的置信水平运行。ANOVA 的结果通过使用带有 95% 置信区间的 ANOVA 图表的形式报告，如图 9 所示。值得注意的是，两个算法的重叠区间表明它们之间没有显著的性能差异。从图 9 可以看出，DIWO 与其他比较算法在每个终止标准上都没有重叠，这意味着 DIWO 与其他算法在 95% 置信水平上的差异是统计显著的。因此，我们可以得出结论，DIWO 在所有终止标准上均显著优于其他比较方法。此外，对于 $\rho = 60$ 和 $\rho = 90$ ，MA 和 P-EDA 的置信区间几乎完全重叠，这表明它们在这些终止标准上具有相似的性能。TPA、HVNS、DE_PLS 和 SVNS_D 在 $\rho = 90$ 时也有重叠的置信区间。

Secondly, the parametric Duncan's multiple range test is employed to grade all compared algorithms into several levels. Table 7 reports the results of grade, where we can see the proposed DIWO algorithm ranks the first level for each termination criterion, i.e., A, and there are no other algorithms in this level, which indicates the differences between DIWO with other compared algorithms are statistically significant and the proposed algorithm achieves the best performance among all compared algorithms. Meanwhile, MA and P-EDA are in the second level, which means they have similar performance. Moreover, DE-ABC is graded into the last level. As also seen from Table 7, the grade of TPA reduces with the increase of running time. For other algorithms, the changes of grade are very small.

其次，参数化的邓肯多重范围测试被用来将所有比较的算法分为几个等级。表 7 报告了分级结果，我们可以看到提出的 DIWO 算法在每个终止准则下都排名第一，即 A，并且这一级别没有其他算法，这表明 DIWO 与其他比较算法的差异在统计学上是显著的，并且提出的算法在所有比较的算法中实现了最佳性能。同时，MA 和 P-EDA 处于第二级别，这意味着它们具有相似的性能。此外，DE-ABC 被划分为最后一个级别。如表 7 所示，TPA 的等级随着运行时间的增加而降低。对于其他算法，等级的变化非常小。

Table 7
表 7
Results of Duncan’ s multiple range test $(\alpha = 0.05)$.
邓肯多重范围测试的结果 $(\alpha = 0.05)$ 。

Rank	$\rho = 30$	$\rho = 60$	$\rho = 90$
A	{DIWO}	{DIWO}	{DIWO}
B	{ MA, P-EDA, TPA }	{MA, P-EDA}	{MA, P-EDA}
C	{SaDIWO}	{SaDIWO, HVNS, TPA}	{SaDIWO}
D	{SVNS_D, DE_PLS, HVNS}	{SVNS_D, DE_PLS}	{HVNS, TPA, DE_PLS}
E	{MFFO}	{MFFO}	{ DE_PLS, TPA }
F	{hmgHS}	{RAIS, hmgHS}	{MFFO}
G	{HDDE}	{HDDE}	{RAIS, hmgHS}
H	{IG}	{IG}	{hmgHS, HDDE}
I	{RAIS}	{DE-ABC}	{IG}
J	{DE-ABC}	-	{DE-ABC}

排名	$\rho = 30$	$\rho = 60$	$\rho = 90$
A	{DIWO}	{DIWO}	{DIWO}
B	{ MA, P-EDA, TPA }	{MA, P-EDA}	{MA, P-EDA}
C	{SaDIWO}	{SaDIWO, HVNS, TPA}	{SaDIWO}
D	{SVNS_D, DE_PLS, HVNS}	{SVNS_D, DE_PLS}	{HVNS, TPA, DE_PLS}
E	{MFFO}	{MFFO}	{ DE_PLS, TPA }
F	{hmgHS}	{RAIS, hmgHS}	{MFFO}
G	{HDDE}	{HDDE}	{RAIS, hmgHS}
H	{IG}	{IG}	{hmgHS, HDDE}
I	{RAIS}	{DE-ABC}	{IG}
J	{DE-ABC}	-	{DE-ABC}

To further statistically justify the performance of the competing algorithms, we also employ a powerful non-parametric statistical method, i.e., Holm's procedure (Holm, 1979) is employed. According to the Refs. Derrac et al. (2011) and García et al. (2009), Holm's procedure is more rigorous than the previous ones and very suitable for comparing evolutionary and swarm intelligence optimization algorithms. In Holm's procedure, we firstly assume that the proposed DIWO (i.e., the control algorithm) is equivalent to other 13 compared algorithms on each termination criterion. Thus, a total of 13 hypotheses are considered for each termination criterion, which are denoted by $H_1, H_2, \ldots, H_{13}$. α is the probability of rejecting at least one H_i that is in fact true. p_i denotes the p -value of H_i . Holm' s procedure orders the p -values of these hypotheses with an ascending order. Then if p -value (i.e., p_i) of H_i is lower than $\alpha/(k-i+1)$, the hypothesis H_i will be rejected, where i is the rank of H_i and k is the total number of hypotheses. The confidence level is set $\alpha = 0.05$. The results of Holm' s procedure are given in Table 8 . As seen in this table, when $\rho = 30$, the hypotheses DIWO=SaDIWO, DIWO=TPA, DIWO=P-EDA and DIWO=MA are accepted, which indicates DIWO is not significantly better than SaDIWO, TPA, P-EDA and MA on this termination criterion. For $\rho = 60$, only P-EDA and MA are not significantly inferior to DIWO. For $\rho = 90$, Holm' s procedure rejects all of 13 pairwise comparisons, which highlights the fact that the proposed DIWO is a better algorithm than other 13 algorithms.

为了进一步从统计学上证明竞争算法的性能，我们还采用了一种强大的非参数统计方法，即 Holm 法 (Holm，1979)。根据 Derrac 等人 (2011) 和 García 等人 (2009) 的文献，Holm 法比之前的方法更为严格，非常适合比较进化算法和群体智能优化算法。在 Holm 法中，我们首先假设提出的 DIWO(即控制算法) 在每个终止准则上与其他 13 种比较的算法等效。因此，对于每个终止准则，我们考虑了总共 13 个假设，表示为 $H_1, H_2, \ldots, H_{13}$ 。α 是拒绝至少一个实际上为真的 H_i 的概率。p_i 表示 H_i 的 p -值。Holm 法将这些假设的 p -值按升序排列。如果 p -值 (即 p_i) 的 H_i 低于 $\alpha/(k-i+1)$ ，则该假设 H_i 将被拒绝，其中 i 是 H_i 的排名，k 是假设的总数。置信水平设置为 $\alpha = 0.05$ 。Holm 法的结果在表 8 中给出。如表格所示，当 $\rho = 30$ 时，接受假设 DIWO=SaDIWO，DIWO=TPA，DIWO=P-EDA 和 DIWO=MA，这表明 DIWO 在此终止准则上并不比 SaDIWO、TPA、P-EDA 和 MA 显著更好。对于 $\rho = 60$ ，只有 P-EDA 和 MA 不显著劣于 DIWO。对于 $\rho = 90$ ，Holm 法拒绝了所有 13 个成对比较，这突显出提出的 DIWO 算法比其他 13 种算法更优。

Although all explanations and details given in the original articles were strictly followed to implement these compared algorithms and closely reproduce the published results, to make the comparison results more convincing and reasonable, we also compare the results of DIWO algorithm with the reported results in the literature. The compared algorithms adopt the same performance evaluation metric (i.e., Eq. (18)) and test instances. These algorithms include HDDE (Wang et al., 2010a), hmgHS (Wang et al., 2011), IG (Ribas et al., 2011), RAIS (Lin and Ying, 2013), TPA (Wang et al., 2012), MA (Pan et al., 2013b), SVNS_S and SVNS_D (Ribas et al., 2013), HVNS (Moslehi and Khorasanian, 2014a), DE_PLS (Tasgetiren et al., 2015), and P-EDA (Shao et al., 2018a). Note that the results of HDDE and MA are obtained from (Tasgetiren et al., 2015), while the results of other algorithms are from their original literature. Additionally, in order to be in consistent with these compared algorithms, the termination criterion of the proposed DIWO algorithm is set $\rho = 100$. Each instance is independently run 10 times. The comparison results are reported in Table 9. As seen from this table, the overall average ARPD of the proposed DIWO algorithm is -0.85, which is superior to the corresponding average ARPD values of other compared algorithms. Probing further into the statistics in Table 9 shows that HDDE, RAIS, DE_PLS, P-EDA, and the proposed DIWO achieve the similar results for the small-scale instances (i.e., 20 jobs), while the results obtained by DE_PLS are slightly better than DIWO for middle scale instances (i.e., 50 jobs). Nevertheless, the DIWO algorithm outperforms other compared algorithms at a considerable margin for large scale instances (i.e., 200 and 500 jobs). Through comparing with the results in the literature, we can further confirm the proposed DIWO algorithm is an effective and efficient approach for BFSP with makespan criterion, especially for the larger scale BFSP. The superiority of the proposed algorithm is mainly attributed to the following aspects. (1)

尽管严格遵循原文中的所有解释和细节来实施这些比较算法并精确重现已发表结果，为了使比较结果

更具说服力和合理性，我们还将 DIWO 算法的结果与文献中报告的结果进行了比较。比较的算法采用了相同的性能评价指标 (即公式 (18)) 和测试实例。这些算法包括 HDDE(Wang et al., 2010a)、hmgHS(Wang et al., 2011)、IG(Ribas et al., 2011)、RAIS(Lin and Ying, 2013)、TPA(Wang et al., 2012)、MA(Pan et al., 2013b)、SVNS_S 和 SVNS_D(Ribas et al., 2013)、HVNS(Moslehi and Khorasanian, 2014a)、DE_PLS(Tasgetiren et al., 2015) 和 P-EDA(Shao et al., 2018a)。注意，HDDE 和 MA 的结果来源于 (Tasgetiren et al., 2015)，而其他算法的结果来自它们原始的文献。此外，为了与这些比较算法保持一致，提出的 DIWO 算法的终止准则设置为 $\rho = 100$ 。每个实例独立运行 10 次。比较结果报告在表 9 中。从表中可以看出，提出的 DIWO 算法的整体平均 ARDP 为-0.85，这优于其他比较算法对应的平均 ARDP 值。进一步探究表 9 中的统计数据表明，HDDE、RAIS、DE_PLS、P-EDA 和提出的 DIWO 在小规模实例 (即 20 个作业) 上取得了类似的结果，而 DE_PLS 在中规模实例 (即 50 个作业) 上得到的结果略优于 DIWO。然而，DIWO 算法在大规模实例 (即 200 和 500 个作业) 上明显优于其他比较算法。通过与文献中的结果进行比较，我们可以进一步确认提出的 DIWO 算法是 BFSP 以最大完工时间标准为准则的有效且高效的方法，尤其是在大规模 BFSP 中。提出算法的优越性主要归因于以下方面。(1)

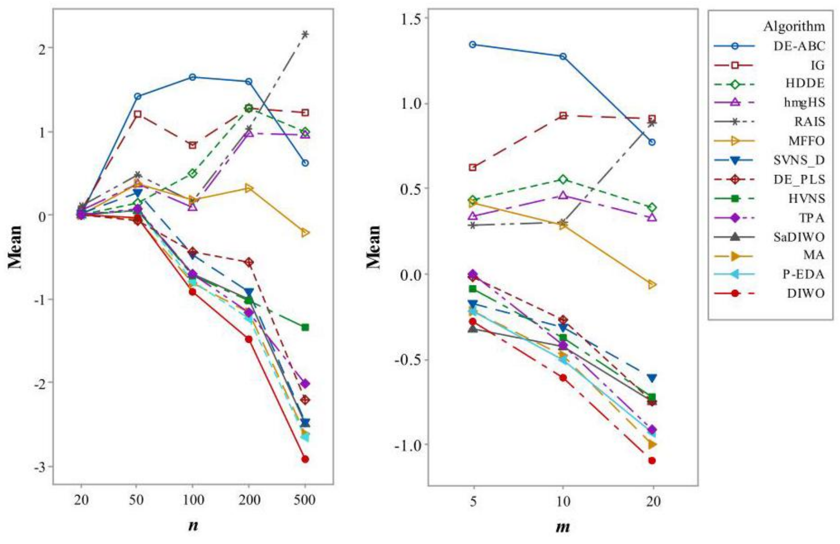

Fig. 7. Interaction plot between algorithm and n with m .
图 7. 算法与 n 以及 m 的交互图。

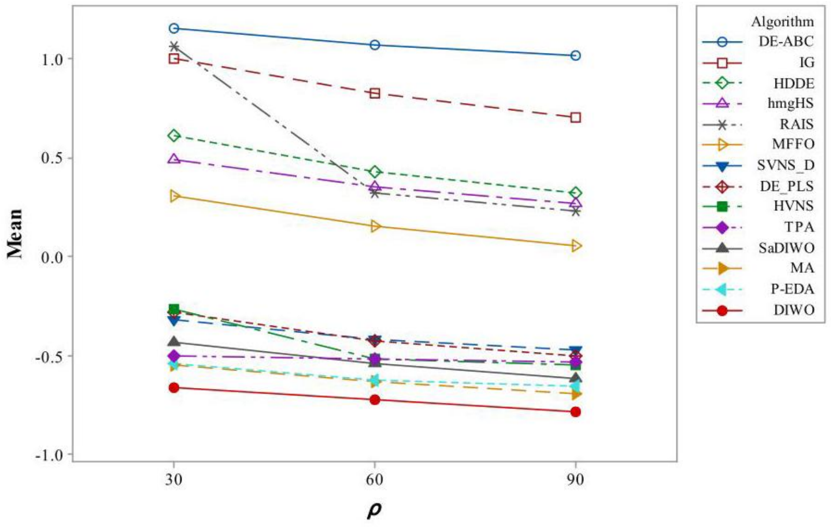

Fig. 8. Interaction plot between algorithm and ρ .
图 8. 算法与 ρ 的交互图。

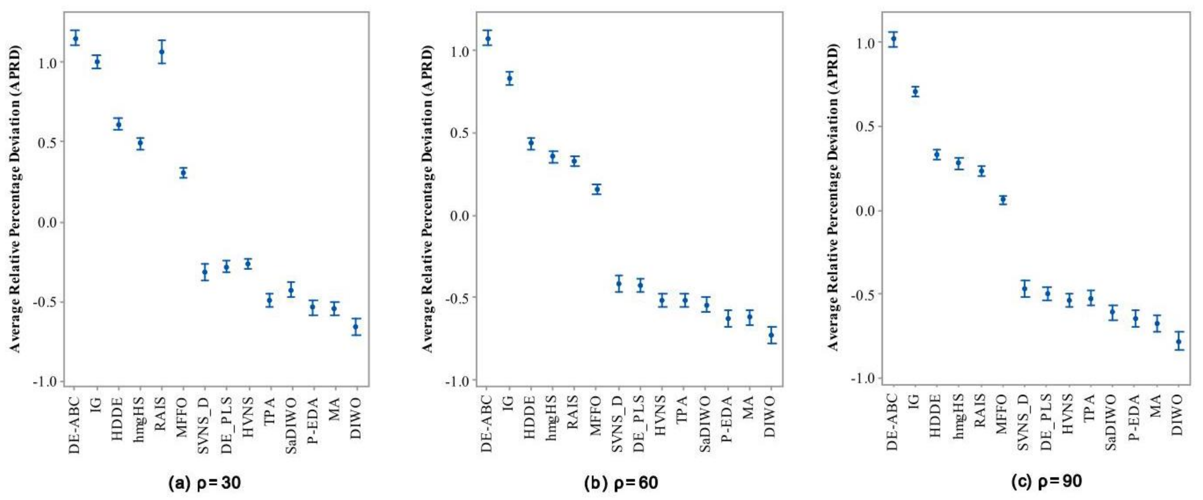

Fig. 9. Means plots and 95% confidence level intervals of all algorithms for different termination criteria.
图 9. 所有算法在不同终止准则下的均值图和 95% 置信水平区间。
Table 8
表 8
The results of Holm’ s procedure $(\alpha = 0.05)$.
Holm 过程 $(\alpha = 0.05)$ 的结果。

i	$\alpha/(k-i+1)$	$\rho = 30$			$\rho = 60$			$\rho = 90$		
		H_i	p_i	R/A	H_i	p_i	R/A	H_i	p_i	R/A
1	0.0039	DIWO = DE-ABC	0.0000	R	DIWO = DE-ABC	0.0000	R	DIWO = DE-ABC	0.0000	R
2	0.0043	DIWO = RAIS	0.0000	R	DIWO = IG	0.0000	R	DIWO = IG	0.0000	R
3	0.0047	DIWO = IG	0.0000	R	DIWO = HDDE	0.0000	R	DIWO = HDDE	0.0000	R
4	0.0051	DIWO = HDDE	0.0000	R	DIWO = hmgHS	0.0000	R	DIWO = hmgHS	0.0000	R
5	0.0057	DIWO = hmgHS	0.0000	R	DIWO = RAIS	0.0000	R	DIWO = RAIS	0.0000	R
6	0.0064	DIWO = MFFO	0.0000	R	DIWO = MFFO	0.0000	R	DIWO = MFFO	0.0000	R
7	0.0073	DIWO = HVNS	0.0001	R	DIWO = SVNS_D	0.0006	R	DIWO = SVNS_D	0.0000	R
8	0.0085	DIWO = DE_PLS	0.0001	R	DIWO = DE_PLS	0.0009	R	DIWO = DE_PLS	0.0005	R
9	0.0102	DIWO = SVNS_D	0.0005	R	DIWO = TPA	0.0199	R	DIWO = TPA	0.0017	R
10	0.0127	DIwo = SaDIwo	0.0190	A	DIWO = HVNS	0.0209	R	DIWO = HVNS	0.0048	R
11	0.0170	DIWO = TPA	0.0980	A	DIWO = SaDIWO	0.0446	R	DIWO = SaDIwo	0.0078	R
12	0.0253	DIWO = P-EDA	0.2145	A	DIWO = P-EDA	0.0603	A	DIWO = P-EDA	0.0168	R
13	0.0500	DIWO = MA	0.2385	A	DIWO = MA	0.1161	A	DIWO = MA	0.0213	R

i	$\alpha/(k-i+1)$	$\rho = 30$			$\rho = 60$			$\rho = 90$		
		H_i	p_i	R/A	H_i	p_i	R/A	H_i	p_i	R/A
1	0.0039	DIWO = DE-ABC	0.0000	R	DIWO = DE-ABC	0.0000	R	DIWO = DE-ABC	0.0000	R
2	0.0043	DIWO = RAIS	0.0000	R	DIWO = IG	0.0000	R	DIWO = IG	0.0000	R
3	0.0047	DIWO = IG	0.0000	R	DIWO = HDDE	0.0000	R	DIWO = HDDE	0.0000	R
4	0.0051	DIWO = HDDE	0.0000	R	DIWO = hmgHS	0.0000	R	DIWO = hmgHS	0.0000	R
5	0.0057	DIWO = hmgHS	0.0000	R	DIWO = RAIS	0.0000	R	DIWO = RAIS	0.0000	R
6	0.0064	DIWO = MFFO	0.0000	R	DIWO = MFFO	0.0000	R	DIWO = MFFO	0.0000	R
7	0.0073	DIWO = HVNS	0.0001	R	DIWO = SVNS_D	0.0006	R	DIWO = SVNS_D	0.0000	R
8	0.0085	DIWO = DE_PLS	0.0001	R	DIWO = DE_PLS	0.0009	R	DIWO = DE_PLS	0.0005	R
9	0.0102	DIWO = SVNS_D	0.0005	R	DIWO = TPA	0.0199	R	DIWO = TPA	0.0017	R
10	0.0127	DIwo = SaDIwo	0.0190	A	DIWO = HVNS	0.0209	R	DIWO = HVNS	0.0048	R
11	0.0170	DIWO = TPA	0.0980	A	DIWO = SaDIWO	0.0446	R	DIWO = SaDIwo	0.0078	R
12	0.0253	DIWO = P-EDA	0.2145	A	DIWO = P-EDA	0.0603	A	DIWO = P-EDA	0.0168	R
13	0.0500	DIWO = MA	0.2385	A	DIWO = MA	0.1161	A	DIWO = MA	0.0213	R

Note: R and A denote reject and accept hypothesis, respectively.
备注:R 和 A 分别表示拒绝和接受假设。
Table 9
表 9
The comparison results of the algorithms.
算法比较结果。

Instance	HDDE	hmgHS	IG	TPA	RAIS	SVNS_S	SVNS_D	HVNS	DE_PLS	MA	P-EDA	DIWO
20 × 5	0.00	0.09	0.39	0.40	0.00	0.12	0.16	0.01	0.00	0.00	0.00	0.00
20 × 10	0.00	0.03	0.48	0.21	0.00	0.15	0.13	0.01	0.00	0.01	0.00	0.00
20 × 20	0.00	0.03	0.31	0.03	0.00	0.08	0.07	0.00	0.00	0.00	0.00	0.00
50 × 5	0.78	0.37	2.71	1.56	-0.19	0.69	0.77	0.17	-0.28	-0.17	-0.11	-0.10
50 × 10	0.53	0.50	3.24	1.07	-0.20	0.88	0.75	-0.06	-0.38	-0.18	-0.24	-0.18
50 × 20	0.35	0.29	2.88	0.76	0.00	0.85	0.70	-0.06	-0.34	-0.20	-0.12	-0.12
100 × 5	1.78	0.80	3.82	1.71	-0.82	-0.19	-0.14	-0.60	-0.81	-0.81	-0.96	-0.98
100 × 10	1.01	0.46	3.34	0.59	-0.88	0.11	0.14	-0.90	-1.25	-1.16	-1.21	-1.19
100 × 20	0.86	0.31	0.03	0.07	-0.71	0.37	0.44	-1.03	-1.14	-1.09	-1.02	-1.17
200 × 10	2.49	1.27	3.85	1.22	-0.33	0.04	0.04	-0.80	-0.93	-0.89	-1.41	-1.44
200 × 20	1.24	0.37	2.31	0.03	-0.63	-0.28	-0.27	-1.50	-0.95	-1.06	-1.82	-1.87
500 × 20	1.54	0.17	1.32	-0.05	-0.02	-1.75	-1.74	-2.00	- 2.62	-2.65	-2.94	-3.08
Average	0.88	0.39	2.31	0.63	-0.32	0.09	0.086	-0.56	-0.73	-0.683	-0.82	-0.85

实例	HDDE	hmgHS	IG 请注意，上述翻译是基于原文直接转换，并未包含对 DIWO、SVNS_D、DE_PLS、TPA、P-EDA、MA、HDDE、hmgHS 和 IG 等专业术语的翻译，因为这些术语可能是特定领域的缩写或专有名词，没有提供足够的上下文来准确翻译。在学术文本中，这类术语通常保持原样。	TPA	RAIS	SVNS_S	SVNS_D	HVNS	DE_PLS	MA	P-EDA	DIWO
20 × 5	0.00	0.09	0.39	0.40	0.00	0.12	0.16	0.01	0.00	0.00	0.00	0.00
20 × 10	0.00	0.03	0.48	0.21	0.00	0.15	0.13	0.01	0.00	0.01	0.00	0.00
20 × 20	0.00	0.03	0.31	0.03	0.00	0.08	0.07	0.00	0.00	0.00	0.00	0.00
50 × 5	0.78	0.37	2.71	1.56	-0.19	0.69	0.77	0.17	-0.28	-0.17	-0.11	-0.10
50 × 10	0.53	0.50	3.24	1.07	-0.20	0.88	0.75	-0.06	-0.38	-0.18	-0.24	-0.18
50 × 20	0.35	0.29	2.88	0.76	0.00	0.85	0.70	-0.06	-0.34	-0.20	-0.12	-0.12
100 × 5	1.78	0.80	3.82	1.71	-0.82	-0.19	-0.14	-0.60	-0.81	-0.81	-0.96	-0.98
100 × 10	1.01	0.46	3.34	0.59	-0.88	0.11	0.14	-0.90	-1.25	-1.16	-1.21	-1.19
100 × 20	0.86	0.31	0.03	0.07	-0.71	0.37	0.44	-1.03	-1.14	-1.09	-1.02	-1.17
200 × 10	2.49	1.27	3.85	1.22	-0.33	0.04	0.04	-0.80	-0.93	-0.89	-1.41	-1.44
200 × 20	1.24	0.37	2.31	0.03	-0.63	-0.28	-0.27	-1.50	-0.95	-1.06	-1.82	-1.87
500 × 20	1.54	0.17	1.32	-0.05	-0.02	-1.75	-1.74	-2.00	- 2.62	-2.65	-2.94	-3.08
平均值	0.88	0.39	2.31	0.63	-0.32	0.09	0.086	-0.56	-0.73	-0.683	-0.82	-0.85

The adopted effective heuristic provides an excellent initial solution. (2) The proposed spatial dispersal operator effectively balances the global exploration and local exploitation. (3) The designed local intensification mechanism effectively exploits the areas around the new created seeds. (4) The adopted competitive exclusion maintains a good population diversity through eliminating the similar weeds in the colony.

采用的有效启发式方法提供了优秀的初始解。(2) 提出的空间分散操作符有效地平衡了全局探索和局部开发。(3) 设计的局部强化机制有效地利用了新创建种子周围的区域。(4) 采用的竞争排斥通过消除群落中的相似杂草，保持了良好的种群多样性。

The excellent performance of the proposed DIWO encourages us to find new best solutions of Taillard's benchmark instances for BFSP with makespan criterion. Most of algorithms for the considered problem have reported their best solutions in their original literature. Thus, we summary these reported results and compare with our proposed DIWO. Some new best solutions are discovered during experiment. Table 10 reports the comparison results, where the best solutions found by DIWO are highlighted in bold. In this table, the 'Source' column represents the methods adopted to obtain the best solution. In the 'Source' column, the numbers from ' 1 ' to ' 15 ' respectively represent HDDE (Wang et al., 2010a), DE-ABC (Han et al., 2015a), MFFO (Han et al., 2016b), IG (Ribas et al., 2011), TPA (Wang et al., 2012), TPA* (Moslehi and Khorasanian, 2014b), RAIS (Lin and Ying, 2013), SVNS (Ribas et al., 2013), MA (Pan et al., 2013a), MA* (Tasgetiren et al., 2015), HVNS (Moslehi and Khorasanian, 2014b), DE_PLS (Tasgetiren et al., 2015), DWWO (Shao et al., 2018b), P-EDA (Shao et al., 2018a), and the proposed DIWO. It should be noted that MA* and TPA* are the extensions of MA and TPA, respectively, which are run more CPU time than the original ones. It can be observed that all compared algorithms obtain the best values for the small-scale instances (i.e., 20 jobs). For the remaining 90 instances, the proposed DIWO provides 47 out of 90 new unique best solutions. Especially for large-scale instances, i.e., $200 \times 10, 200 \times 20$, and 500×20 , the proposed DIWO almost updates all of best solutions. The comparison results above further demonstrate that the proposed DIWO has good performance on solving the considered problem. Additionally, we also provide the sequences of the new best solutions on the online supplementary material, which can be a reference for future work along this line of research.

所提出 DIWO 的卓越表现鼓励我们去寻找 Taillard 基准实例在 BFSP 中以最大完工时间标准的新最佳解。大多数针对所考虑问题的算法都在其原始文献中报告了它们的最佳解。因此，我们总结了这些报告的结果并与我们提出的 DIWO 进行比较。在实验过程中发现了一些新的最佳解。表 10 报告了比较结果，其中 DIWO 找到的最佳解以粗体突出显示。在此表中，“来源”列表示用于获得最佳解的方法。在“来源”列中，从'1' 到'15' 的数字分别代表 HDDE (Wang et al., 2010a)、DE-ABC (Han et al., 2015a)、MFFO (Han et al., 2016b)、IG (Ribas et al., 2011)、TPA (Wang et al., 2012)、TPA* (Moslehi and Khorasanian, 2014b)、RAIS (Lin and Ying, 2013)、SVNS (Ribas et al., 2013)、MA (Pan et al., 2013a)、MA* (Tasgetiren et al., 2015)、HVNS (Moslehi and Khorasanian, 2014b)、DE_PLS (Tasgetiren et al., 2015)、DWWO (Shao et al., 2018b)、P-EDA (Shao et al., 2018a) 以及所提出的 DIWO。应注意 MA* 和 TPA* 分别是 MA 和 TPA 的扩展，它们比原算法运行了更多的 CPU 时间。可以观察到，所有比较的算法对于小规模实例 (即 20 个作业) 都获得了最佳值。对于剩余的 90 个实例，所提出的 DIWO 提供了 90 个中的 47 个新的独特最佳解。特别是对于大规模实例，即 $200 \times 10, 200 \times 20$ 和 500×20 ，所提出的 DIWO 几乎更新了所有的最佳解。上述比较结果进一步证明了所提出的 DIWO 在解决所考虑问题上的良好性能。此外，我们还在在线辅助材料中提供了新最佳解的序列，这可以作为未来沿此研究方向的参考。

6. Conclusions and future research

6. 结论与未来研究

In this paper, we propose a discrete invasive weed optimization for solving the blocking flow-shop scheduling problem with makespan criterion. In the proposed algorithm, an effective heuristic and the random method were combined to generate an initial plant population with high quality and diversity. A random-insertion-based spatial dispersal was developed by means of the normal distribution to guide the global exploration and local exploitation. A shuffle-based referenced local search was incorporated to strengthen local exploitation. An improved competitive exclusion was employed to guarantee the offspring population not only has good quality but also a certain level of diversity. The parameters setting was investigated by using a design of experiments approach. The effectiveness of the proposed spatial dispersal and local search was demonstrated by numerical comparisons. To verify the efficiency and effectiveness of the proposed algorithm, an extensive comparison was carried out. The comparison results showed that the proposed DIWO algorithm outperformed the recently published algorithms in terms of solution quality and search efficiency. Moreover, we have also updated 47 best solutions out of 120 Taillard's benchmark instances.

在本文中，我们提出了一种离散的入侵杂草优化算法，用于解决以最大完工时间为标准的阻塞流水车间调度问题。在提出的算法中，有效的启发式方法和随机方法相结合，以生成高质量和多样性的初始种群。通过正态分布，开发了一种基于随机插入的空间扩散方法，以引导全局探索和局部开发。引入了一种基于洗牌的参考局部搜索方法，以加强局部开发。采用改进的竞争排斥策略，以确保子代种群不仅具有良好的质量，而且具有一定的多样性。通过实验设计方法研究了参数设置。通过数值比较证明了提出的空间扩散和局部搜索的有效性。为了验证所提算法的效率和有效性，进行了广泛的比较。比较结果表明，提出的 DIWO 算法在解的质量和搜索效率方面优于近期发表的算法。此外，我们还更新了 120 个 Taillard 基准实例中的 47 个最佳解决方案。

For the future research, the DIWO algorithm will be extended to solve the BFSP with other objectives, such as total flow time, total tardiness time and multi-objective BFSP. Additionally, we will try designing some adaptive ways to reduce the number of parameters for DIWO algorithm to further enhance the performance of it, and then apply it to solve more realistic scheduling problems.

对于未来的研究，DIWO 算法将被扩展以解决具有其他目标的 BFSP，例如总流程时间、总延迟时间和多目标 BFSP。此外，我们将尝试设计一些自适应方法来减少 DIWO 算法的参数数量，以进一步提高其性能，并将其应用于解决更实际的调度问题。

Table 10

表 10

Best solutions for BFSP with makespan criterion.

以最大完工时间为标准的 BFSP 最佳解决方案。

Instance	Best	Source	DIWO	Instance	Best	Source	DIWO	Instance	Best	Source	DIWO
20 × 5				50 × 10				100 × 20			
Ta1	1374	1–15	1374	Ta41	3611	10	3614	Ta81	7709	15	7709
Ta2	1408	1–15	1408	Ta42	3470	14	3473	Ta82	7 744	14	7 769
Ta3	1280	1–15	1280	Ta43	3465	15	3465	Ta83	7723	14	7725
Ta4	1448	1–15	1448	Ta44	3650	8	3654	Ta84	7 743	14	7 755
Ta5	1341	1–15	1341	Ta45	3582	8	3619	Ta85	7 730	13	7 744
Ta6	1363	1–15	1363	Ta46	3571	14	3575	Ta86	7 779	14	7785
Ta7	1381	1–15	1381	Ta47	3667	14	3669	Ta87	7 857	15	7 857
Ta8	1379	1–15	1379	Ta48	3549	15	3549	Ta88	7 898	11	7 902
Ta9	1373	1–15	1373	Ta49	3508	8	3518	Ta89	7818	14	7835
Ta10	1283	1–15	1283	Ta50	3608	12	3610	Ta90	7 842	15	7 842
20 × 10				50 × 20				200 × 10			
Ta11	1698	1–15	1698	Ta51	4479	15	4479	Ta91	13 149	14	13 166
Ta12	1833	1–15	1833	Ta52	4262	14, 15	4262	Ta92	13085	15	13085
Ta13	1659	1–15	1659	Ta53	4261	12, 15	4261	Ta93	13 183	15	13 183
Ta14	1535	1–15	1535	Ta54	4339	12	4342	Ta94	13097	15	13 097
Ta15	1617	1–15	1617	Ta55	4249	12	4250	Ta95	13100	15	13 100
Ta16	1590	1–15	1590	Ta56	4271	12, 15	4271	Ta96	12883	15	12883
Ta17	1622	1–15	1622	Ta57	4291	13, 14	4296	Ta97	13408	15	13 408
Ta18	1731	1–15	1731	Ta58	4298	14, 15	4298	Ta98	13266	15	13 266
Ta19	1747	1–15	1747	Ta59	4304	12	4305	Ta99	13080	15	13080
Ta20	1782	1–15	1782	Ta60	4398	15	4398	Ta100	13 193	15	13 193
20 × 20				100 × 5				200 × 20			
Ta21	2436	1–15	2436	Ta61	6065	15	6065	Ta101	14 192	9	14516
Ta22	2234	1–15	2234	Ta62	5934	15	5934	Ta102	14714	15	14714
Ta23	2479	1–15	2479	Ta63	5851	8	5876	Ta103	14831	15	14831
Ta24	2348	1–15	2348	Ta64	5656	9	5684	Ta104	14801	15	14801
Ta25	2435	1–15	2435	Ta65	5896	15	5896	Ta105	14609	15	14609
Ta26	2383	1–15	2383	Ta66	5755	15	5755	Ta106	14737	15	14737
Ta27	2390	1–15	2390	Ta67	5915	15	5915	Ta107	14770	15	14770
Ta28	2328	1–15	2328	Ta68	5809	14	5825	Ta108	14823	15	14823
Ta29	2363	1–15	2363	Ta69	6027	15	6027	Ta109	14691	15	14691
Ta30	2323	1–15	2323	Ta70	6059	14, 15	6059	Ta110	14711	15	14711
50 × 5				100 × 10				500 × 20			
Ta31	2980	11	2983	Ta71	6906	15	6906	Ta111	35380	15	35 380
Ta32	3180	15	3180	Ta72	6656	15	6656	Ta112	35736	15	35736
Ta33	2995	14	3000	Ta73	6797	14	6807	Ta113	35 406	15	35 406
Ta34	3115	15	3115	Ta74	7035	15	7035	Ta114	35 030	13	35738
Ta35	3139	8	3145	Ta75	6728	15	6728	Ta115	35 417	15	35 417
Ta36	3158	13	3161	Ta76	6537	14	6549	Ta116	35740	15	35740
Ta37	3005	11	3008	Ta77	6689	15	6689	Ta117	35299	15	35 299
Ta38	3042	11	3044	Ta78	6746	14,15	6746	Ta118	35515	15	35515
Ta39	2889	13,14	2891	Ta79	6928	10	6958	Ta119	35268	15	35268
Ta40	3097	14	3102	Ta80	6855	11	6881	Ta120	35609	15	35609

实例	最佳	来源	DIWO(自己动手做)	实例	最佳	来源	DIWO(自己动手做)	实例	最佳	来源	DIWO(自己动手做)
20 × 5				50 × 10				100 × 20			
Ta1	1374	1–15	1374	Ta41	3611	10	3614	Ta81	7709	15	7709
Ta2	1408	1–15	1408	Ta42	3470	14	3473	Ta82	7 744	14	7 769
Ta3	1280	1–15	1280	Ta43	3465	15	3465	Ta83	7723	14	7725
Ta4	1448	1–15	1448	Ta44	3650	8	3654	Ta84	7 743	14	7 755
Ta5	1341	1–15	1341	Ta45	3582	8	3619	Ta85	7 730	13	7 744
Ta6	1363	1–15	1363	Ta46	3571	14	3575	Ta86	7 779	14	7785
Ta7	1381	1–15	1381	Ta47	3667	14	3669	Ta87	7 857	15	7 857
Ta8	1379	1–15	1379	Ta48	3549	15	3549	Ta88	7 898	11	7 902
Ta9	1373	1–15	1373	Ta49	3508	8	3518	Ta89	7818	14	7835
Ta10	1283	1–15	1283	Ta50 如果这些标识符实际上代表特定的学术文本，请提供具体的文本内容，以便进行翻译。	3608	12	3610	Ta90	7 842	15	7 842
20 × 10				50 × 20				200 × 10			
Ta11	1698	1–15	1698	Ta51	4479	15	4479	Ta91	13 149	14	13 166
Ta12	1833	1–15	1833	Ta52	4262	14, 15	4262	Ta92	13085	15	13085
Ta13	1659	1–15	1659	Ta53	4261	12, 15	4261	Ta93	13 183	15	13 183
Ta14	1535	1–15	1535	Ta54	4339	12	4342	Ta94	13097	15	13 097
Ta15	1617	1–15	1617	Ta55	4249	12	4250	Ta95	13100	15	13 100
Ta16	1590	1–15	1590	Ta56	4271	12, 15	4271	Ta96	12883	15	12883
Ta17	1622	1–15	1622	Ta57	4291	13, 14	4296	Ta97 请注意，因为没有具体的文本内容，所以"translation" 字段只是复制了原来的"text" 字段内容。在实际应用中，您需要提供具体的文本内容才能进行准确的翻译。	13408	15	13 408
Ta18	1731	1–15	1731	Ta58	4298	14, 15	4298	Ta98	13266	15	13 266
Ta19	1747	1–15	1747	Ta59	4304	12	4305	Ta99	13080	15	13080
Ta20	1782	1–15	1782	Ta60	4398	15	4398	Ta100	13 193	15	13 193
20 × 20				100 × 5				200 × 20			
Ta21	2436	1–15	2436	Ta61	6065	15	6065	Ta101	14 192	9	14516
Ta22	2234	1–15	2234	Ta62	5934	15	5934	Ta102	14714	15	14714
Ta23	2479	1–15	2479	Ta63	5851	8	5876	Ta103	14831	15	14831
Ta24	2348	1–15	2348	Ta64	5656	9	5684	Ta104	14801	15	14801
Ta25 由于输入的 YAML 文档中的"text" 字段包含的是"Ta24"、"Ta64"、"Ta104" 和"Ta25" 这样的代码或标识符，而不是学术文本，因此翻译结果保留了原样。如果这些标识符实际上代表特定的学术文本，请提供具体的文本内容以便进行翻译。	2435	1–15	2435	Ta65	5896	15	5896	Ta105	14609	15	14609
Ta26	2383	1–15	2383	Ta66	5755	15	5755	Ta106	14737	15	14737
Ta27	2390	1–15	2390	Ta67	5915	15	5915	Ta107	14770	15	14770
Ta28	2328	1–15	2328	Ta68	5809	14	5825	Ta108 如果这些字符串实际上是某种代码或者有特定的学术含义，请提供更多的上下文，以便进行准确的翻译。	14823	15	14823
Ta29	2363	1–15	2363	Ta69	6027	15	6027	Ta109	14691	15	14691
Ta30	2323	1–15	2323	Ta70	6059	14, 15	6059	Ta110	14711	15	14711
50 × 5				100 × 10				500 × 20			
Ta31	2980	11	2983	Ta71	6906	15	6906	Ta111	35380	15	35 380
Ta32	3180	15	3180	Ta72	6656	15	6656	Ta112	35736	15	35736
Ta33	2995	14	3000	Ta73	6797	14	6807	Ta113	35 406	15	35 406
Ta34	3115	15	3115	Ta74	7035	15	7035	Ta114 由于输入的 YAML 文档中的"text" 字段包含的文本"Ta113"、"Ta34"、"Ta74" 和"Ta114" 没有提供具体的学术内容，因此无法进行翻译。如果这些是特定的学术代码或术语，且没有提供上下文，保留原样是合适的做法。	35 030	13	35738
Ta35	3139	8	3145	Ta75	6728	15	6728	Ta115	35 417	15	35 417
Ta36	3158	13	3161	""	6537	14	6549	""	35740	15	35740
""	3005	11	3008	""	6689	15	6689	""	35299	15	35 299
""	3042	11	3044	""	6746	14,15	6746	""	35515	15	35515
""	2889	13,14	2891	""	6928	10	6958	"" 如果这些"Ta" 标识符实际上是某种编码的学术文本，需要提供解码方法或文本的实际含义才能进行翻译。	35268	15	35268
Ta40	3097	14	3102	Ta80	6855	11	6881	Ta120	35609	15	35609

Acknowledgments

致谢

This research is supported by the National Natural Science Foundation of China [grant number U1433116]; the Fundamental Research Funds for the Central Universities [grant number NP2017208]; the Funding of Jiangsu Innovation Program for Graduate Education [grant number KYLX16_0382]; and the Postgraduate Research & Practice Innovation Program of Jiangsu Province [grant number KYCX17_0287].

本项研究得到了中国国家自然科学基金的支持 (项目编号 U1433116)；中央高校基本科研业务费的支持 (项目编号 NP2017208)；江苏省研究生教育创新计划基金的支持 (项目编号 KYLX16_0382)；以及江苏省研究生科研与实践创新计划项目的支持 (项目编号 KYGX17_0287)。

Appendix A. Supplementary data

附录 A. 补充数据

Supplementary material related to this article can be found online at
与本文相关的补充材料可以在网上找到 https://doi.org/10.1016/j.engappai.2018.11.005.

References

参考文献

Barisal, A.K., Prusty, R.C., 2015. Large scale economic dispatch of power systems using oppositional invasive weed optimization. Appl. Soft Comput. 29, 122-137.

Basak, A., Maity, D., Das, S., 2013. A differential invasive weed optimization algorithm for improved global numerical optimization. Appl. Math. Comput. 219, 6645-6668.

Črepinšek, M., Liu, S.H., Mernik, M., 2014. Replication and comparison of computational experiments in applied evolutionary computing: Common pitfalls and guidelines to avoid them. Appl. Soft Comput. 19, 161-170.

Dastranj, A., 2017. Optimization of a Printed UWB Antenna: Application of the invasive weed optimization algorithm in antenna design. IEEE Antennas Propag. Mag. 59, 48- 57.

Davendra, D., BialicDavendra, M., 2013. Scheduling flow shops with blocking using a discrete self-organising migrating algorithm. Int. J. Prod. Res. 51, 2200-2218.

Davendra, D., Zelinka, I., Bialic-Davendra, M., Senkerik, R., Jasek, R., 2012. Clustered enhanced differential evolution for the blocking flow shop scheduling problem. CEJOR Cent. Eur. J. Oper. Res. 20, 679-717.

Deng, G.L., Zhang, S.N., Zhao, M., 2016. A discrete group search optimizer for blocking flow shop multi-objective scheduling. Adv. Mech. Eng. 8, 1-9.

Derrac, J., García, S., Molina, D., Herrera, F., 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and

Ding, J.-Y., Song, S., Gupta, J.N.D., Wang, C., Zhang, R., Wu, C., 2016. New block properties for flowshop scheduling with blocking and their application in an iterated greedy algorithm. Int. J. Prod. Res. 54, 4759-4772.

Eddaly, M., Jarboui, B., Siarry, P., 2016. Combinatorial particle swarm optimization for solving blocking flowshop scheduling problem. J. Comput. Des. Eng. 3, 295-311.

Fernandez-Viagas, V., Leisten, R., Framinan, J.M., 2016. A computational evaluation of constructive and improvement heuristics for the blocking flow shop to minimise total flowtime. Expert Syst. Appl. 61, 290-301.

García, S., Molina, D., Lozano, M., Herrera, F., 2009. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Rea Parameter Optimization. J. Heuristics 15, 617–644. Comput. Oper. Res. 37, 960-969.

Grabowski, J., Pempera, J., 2000. Sequencing of jobs in some production system. European J. Oper. Res. 125, 535-550.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey. Ann. Discrete Math. 5, 287-326.

Hall, N.G., Sriskandarajah, C., 1996. A survey of machine scheduling problems with blocking and no-wait in process. Oper. Res. 44, 510-525.

Han, Y., Gong, D., Li, J., Zhang, Y., 2016a. Solving the blocking flow shop scheduling problem with makespan using a modified fruit fly optimisation algorithm. Int. J. Prod. Res. 54, 1-16.

Han, Y., Gong, D., Li, J., Zhang, Y., 2016b. Solving the blocking flow shop scheduling problem with makespan using a modified fruit fly optimisation algorithm. Int. J. Prod. Res. 54, 6782-6797.

Han, Y.Y., Gong, D., Sun, X., 2015a. A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking. Eng. Optim. 47, 1-20.

Han, Y.Y., Pan, Q.K., Li, J.Q., Sang, H.Y., 2011. An improved artificial bee colony algorithm for the blocking flowshop scheduling problem. Int. J. Adv. Manuf. Technol. 60, 1149 − 1159 .

Han, Y.Y., Sun, D.G., Xiaoyan, 2015b. A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking. Eng. Holm, S., 1979. A simple sequentially rejective multiple test procedure. Scand. J. Stat. 6, 65–70.

Jarboui, B., Eddaly, M., Siarry, P., Rebaï, A., 2009. An estimation of distribution algorithm for minimizing the makespan in blocking flowshop scheduling problems. In: Chakraborty, U.K. (Ed.), Computational Intelligence in Flow Shop and Job Shop Scheduling. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 151-167.

Karabulut, K., 2016. A hybrid iterated greedy algorithm for total tardiness minimization in permutation flowshops. Comput. Ind. Eng. 98, 300-307.

Koren, Y., Wang, W., Gu, X., 2017. Value creation through design for scalability of reconfigurable manufacturing systems. Int. J. Prod. Res. 55, 1227-1242.

Li, X., Wang, Q., Wu, C., 2009. Efficient composite heuristics for total flowtime minimization in permutation flow shops. Omega 37, 155-164.

Liang, J.J., Pan, Q.K., Chen, T., Wang, L., 2011. Solving the blocking flow shop scheduling problem by a dynamic multi-swarm particle swarm optimizer. Int. J. Adv. Manuf. Technol. 55, 755-762.

Lin, S.W., Ying, K.C., 2013. Minimizing makespan in a blocking flowshop using a revised artificial immune system algorithm. Omega 41, 383-389.

Mccormick, S.T., Pinedo, M.L., Shenker, S., Wolf, B., 1989. Sequencing in an assembly line with blocking to minimize cycle time. Oper. Res. 37, 925-935.

Merchan, A.F., Maravelias, C.T., 2016. Preprocessing and tightening methods for time-indexed MIP chemical production scheduling models. Comput. Chem. Eng. 84, 516- 535.

Montgomery, D.C., 2008. Design and Analysis of Experiments. John Wiley & Sons.

Moslehi, G., Khorasanian, D., 2014a. A hybrid variable neighborhood search algorithm for solving the limited-buffer permutation flow shop scheduling problem with the makespan criterion. Comput. Oper. Res. 52 (Part B), 260-268.

Moslehi, G., Khorasanian, D., 2014b. A hybrid variable neighborhood search algorithm for solving the limited-buffer permutation flow shop scheduling problem with the makespan criterion. Comput. Oper. Res. 52, 260-268.

Nagano, M.S., Komesu, A.S., Miyata, H.H., 2017. An evolutionary clustering search for the total tardiness blocking flow shop problem. J. Intell. Manuf. 1-15.

Niknamfar, A.H., Niaki, S.T.A., 2018. A binary-continuous invasive weed optimization algorithm for a vendor selection problem. Knowl.-Based Syst. 140, 158-172.

Nouri, N., Ladhari, T., 2017. Evolutionary multiobjective optimization for the multimachine flow shop scheduling problem under blocking. Ann. Oper. Res. 1-18.

Pan, Q.-K., Ruiz, R., 2014. An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem. Omega 44, 41-50.

Pan, Q.-K., Wang, L., Sang, H.-Y., Li, J.-Q., Liu, M., 2013a. A high performing memetic algorithm for the flowshop scheduling problem with blocking. IEEE Trans. Automat. Sci. Eng. 10, 741-756.

Pan, Q.K., Wang, L., Sang, H.Y., Li, J.Q., Liu, M., 2013b. A high performing memetic algorithm for the flowshop scheduling problem with blocking. IEEE Trans. Automat. Sci. Eng. 10, 741-756.

Pinedo, M.L., 2012. Scheduling: Theory, Algorithms, and Systems. Springer US.

Rama Prabha, D., Jayabarathi, T., 2016. Optimal placement and sizing of multiple distributed generating units in distribution networks by invasive weed optimization algorithm. Ain Shams Eng. J. 7, 683-694.

Rezaei Pouya, A., Solimanpur, M., Jahangoshai Rezaee, M., 2016. Solving multi-objective portfolio optimization problem using invasive weed optimization. Swarm Evol. Comput. 28, 42-57.

Riahi, V., Khorramizadeh, M., Hakim Newton, M.A., Sattar, A., 2017. Scatter search for mixed blocking flowshop scheduling. Expert Syst. Appl. 79, 20-32.

Ribas, I., Companys, R., Tort-Martorell, X., 2011. An iterated greedy algorithm for the flowshop scheduling problem with blocking. Omega 39, 293-301.

Ribas, I., Companys, R., Tort-Martorell, X., 2013. A competitive variable neighbourhood search algorithm for the blocking flow shop problem. Eur. J. Ind. Eng. 7, 729-754.

Ribas, I., Companys, R., Tort-Martorell, X., 2015. An efficient Discrete Artificial Bee Colony algorithm for the blocking flow shop problem with total flowtime minimization. Ronconi, D.P., 2004. A note on constructive heuristics for the flowshop problem with blocking. Int. J. Prod. Econ. 87, 39-48.

Roy, G.G., Das, S., Chakraborty, P., Suganthan, P.N., 2011. Design of non-uniform circular antenna arrays using a modified invasive weed optimization algorithm. IEEE Trans. Antennas Propag. 59, 110-118.

Sang, H.-Y., Duan, P.-Y., Li, J.-Q., 2018a. An effective invasive weed optimization algorithm for scheduling semiconductor final testing problem. Swarm Evol. Comput. 38, $42-53$.

Sang, H.-Y., Pan, Q.-K., Duan, P.-Y., Li, J.-Q., 2018b. An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems. J. Intell. Manuf. 29, 1337-1349.

Saravanan, B., Vasudevan, E.R., Kothari, D.P., 2014. Unit commitment problem solution using invasive weed optimization algorithm. Int. J. Electr. Power Energy Syst. 55, 21-28.

Shao, W., Pi, D., 2016. A self-guided differential evolution with neighborhood search for permutation flow shop scheduling. Expert Syst. Appl. 51, 161-176.

Shao, Z., Pi, D., Shao, W., 2017. Self-adaptive discrete invasive weed optimization for the blocking flow-shop scheduling problem to minimize total tardiness. Comput. Ind. Eng. $111, 331-351$.

Shao, Z., Pi, D., Shao, W., 2018b. A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times. Swarm Evol. Comput. 40, 1-23.

Taillard, E., 1993. Benchmarks for basic scheduling problems. European J. Oper. Res. 64, 278-285.

Tasgetiren, M.F., Kizilay, D., Pan, Q.-K., Suganthan, P.N., 2017. Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. Comput. Oper. Res. 77, 111-126.

Tasgetiren, M.F., Pan, Q.K., Kizilay, D., Suer, G., 2015. A populated local search with differential evolution for blocking flowshop scheduling problem. In: 2015 IEEE Congress on Evolutionary Computation (CEC). IEEE, Sendai, Japan, pp. 2789-2796.

Wang, L., Pan, Q.-K., Suganthan, P.N., Wang, W.-H., Wang, Y.-M., 2010a. A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. Comput. Oper. Res. 37, 509-520.

Wang, L., Pan, Q.K., Tasgetiren, M.F., 2010b. Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms. Expert Syst. Appl. 37, 7929-7936.

Wang, L., Pan, Q.K., Tasgetiren, M.F., 2011. A hybrid harmony search algorithm for the scheduling with blocking to minimize makespan. Comput. Oper. Res. 39, 2880-2887.

Wang, X., Tang, L., 2012. A discrete particle swarm optimization algorithm with self-adaptive diversity control for the permutation flowshop problem with blocking. Appl. Soft Comput. 12, 652-662.

Zheng, Z.-x., Li, J.-q., 2018. Optimal chiller loading by improved invasive weed optimization algorithm for reducing energy consumption. Energy Build. 161, 80-88.

Zhou, Y., Luo, Q., Chen, H., He, A., Wu, J., 2015. A discrete invasive weed optimization algorithm for solving traveling salesman problem. Neurocomputing 151 (Part 3), 1227-1236.