Hybrid Coral Reef Optimization Algorithm Employed Local Search Technique for Job Shop Scheduling Problems

Chin-Shiuh Shieh[1], Thanh-Tuan Nguyen[1,2(✉)], Dinh-Cuong Nguyen[1], Thanh-Nghia Nguyen[3], Mong-Fong Horng[1], and Denis Miu[4]

[1] Department of Electronic Engineering, National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan
tuannt@ntu.edu.vn
[2] Department of Electronics and Automation Engineering, Nha Trang University, Nha Trang, Vietnam
[3] FPT University HCMC, Ho Chi Minh City, Vietnam
[4] Genie Networks, Taipei, Taiwan

Abstract. The JSSP (job shop scheduling problem) is a crucial problem in operational research with certain real-world applications. Due to the fact that the JSSP is an NP-hard (nondeterministic polynomial time) issue, approximation techniques are frequently employed to solve it. This paper introduces a novel biologically-inspired metaheuristic algorithm called Coral Reef Optimization (CRO) in combination with local search strategies Simulated Annealing (SA) significantly improves performance and solution-finding speed. The performance of hybrid algorithms is examined by solving various instances of JSSP. The results indicate that local search methods greatly improve the search efficiency of the hybrid algorithm in comparison to the original algorithm, which was used to assess the improvement. Moreover, comparative findings with five state-of-the-art algorithms from the literature demonstrate that the proposed hybrid algorithms have advantageous search capabilities.

Keywords: Job-shop scheduling · coral reef optimization · local search · hybrid approach

1 Introduction

Scheduling production is essential in product manufacture and directly impacts system efficiency and overall manufacturing productivity [1]. In several businesses, production scheduling has become a significant challenge. To enhance production efficiency, dozens of innovative strategies are being evaluated, with an emphasis on schedule optimization [2]. Consequently, both the scientific and industry communities are focusing on production schedule optimization [3]. Since the 1950s, industrialization has considered production schedules as an important topic, and the job shop scheduling problem

R. Hou et al. (Eds.): BDTA 2021/2022, LNICST 480, pp. 213–231, 2023.
https://doi.org/10.1007/978-3-031-33614-0_15

(JSSP) is a fundamental production scheduling model [4]. Since Johnson's (1954) first two-machine scheduling system [5] the JSSP's complexity has expanded in direct proportion to the number of devices and workloads. The JSSP is classified as NP-hard (non-deterministic polynomial time) due to its tremendous complexity [6]. Solving large-scale JSSP in a sensible time frame has been studied for decades. JSSP is evolving into a variety of new forms with distinct features and characteristics for the rising workloads. It responds in various ways to variations in the fundamental JSSP [7]. According to a study by Xiong et al. [8], hundreds of studies and combinatorial optimization of individual aspects of JSSP and its implementations in the stated sectors were conducted and provided throughout five years period from 2016 to 2021.

In computer science and operations research, JSSP is classified as a multi-stage, static, deterministic task scheduling issue, and its solution strategies vary at each level of research development. The basis of JSSP consists of:

- A set of n jobs $J = \{J_i | i = 1, 2, ...n\}$, where J_i denotes ith job $(1 \leq i \leq n)$.
- A set of m machines $M = \{M_j | j = 1, 2, ...m\}$, and Mj denotes jth machine.
- Each job Ji has a specific set of operations $O = \{O_{i1}, O_{i2}, ..., O_{ik}\}$, where k is the total operations in job Ji.
- Operation O_{ij} will be processed only once the operation O_{ij-1} has been completed in job Ji.

Scheduling distributes shared resources to concurrent tasks during processing. The system entails allocating and structuring restricted resources according to the problem's limits, such as activity sequence and time consumption, and provides a strategy for reaching optimization goals. The following are the fundamental JSSP requirements [9]:

- Each operation is executed separately from the others.
- No job operation can commence until all prior operations are finished.
- Once a processing operation has started, it will not be halted until the process is completed.
- It is impossible to perform numerous operations of the same work at one time.
- Job operations must queue until the next suitable machine becomes available.
- Single machine can only carry out one operation at a time.
- The machine will be idle throughout the unallocated period.

Notably, the set of constraints in real-world problems is more complex, including multiple objective scheduling issues in a job shop, processing times that can be either deterministic or probabilistic, and idle time requests that are restricted to no more than two subsequent machines or none at all. Any adjustment to the problem's constraints might result in a new problem variant. Therefore, JSSP problem-solving procedures differ with each study development stage [10]. One of the practical techniques to JSSP, metaheuristic optimization can deliver a good optimization solution in a sensible amount of time [11]. Enabling the speedy creation of high-quality solutions, metaheuristic algorithms apply unique search techniques to explore the solution space and avoid becoming trapped in local optima by directing the viable solution with a bias. Modern metaheuristic algorithms additionally use several mathematical models [12] and analytical operating processes [13] to improve performance.

The coral reef optimization approach (CRO) is one of the complicated bio-inspired computer techniques utilized to address engineering and scientific issues by modeling the "formation" and "reproduction" of corals in coral reefs. In 2014, Salcedo-Sanz et al. were the first to propose the method [14]. Since then, it has been applied to a number of relevant problems, such as optimum mobile network deployment [15], improved battery scheduling of microgrids [16], and wind speed prediction systems with success in "Offshore Wind Farm Design" [17]. This paper contributes by offering improved coral reef optimization strategies using local search techniques for the JSSP function. On the basis of the original coral reef optimization (CRO) approach, a hybrid algorithm, CROLS, which combines CRO with the Simulated Annealing (SA) strategy methodology, has been created and presented. Moreover, comparative findings with five state-of-the-art algorithms from the literature to demonstrate that the proposed hybrid algorithms have advantageous search capabilities.

The rest of this paper is organized as follows: Sect. 2 provides a summary of related work. Section 3 describes the rationale for the proposed method. The experimental results are described in Sect. 4. Section 5 concludes this research.

2 Related Work

Since its beginning, operation research has emphasized accurate techniques for tackling combinatorial problems with numerous variables. Exact algorithms are defined as those that ensure correct optimization problem answers. Almost every constrained combinatorial optimization problem might be solved using precise algorithms by finding all feasible solutions in a short amount of time [18]. However, it has been argued that when precise algorithms are used to solve combinatorial optimization problems, the time necessary to determine the optimal method increases exponentially with the problem's complexity. Branch and bind algorithms and mixed integer programming are the most often used accurate algorithms for tackling JSSP problems [19]. Small-scale JSSP seldom depicts production environments in actual production; thus, it is essential to analyze more complicated issues involving several works and resources. Due to resource constraints and lengthy execution periods, however, the exact approaches are seldom applied to large-scale scenarios.

Numerous approaches based on artificial intelligence were first suggested and opened a new area in the study of problem-solving strategies [20], and approximation algorithms are one of the most investigated solutions for large-scale combinatorial optimization problems. Although approximation algorithms are not guaranteed to uncover an ideal solution, they are guaranteed to identify a near-optimal solution in a reasonable period of time. Consequently, it has become a new area of study for addressing complicated and large-scale issues. There are two sorts of approximation algorithms: heuristic algorithms and metaheuristic algorithms [21].

Heuristic methods exist in two regions: constructive and local search strategies [22]. In normal constructive algorithms, solutions are constructed piecemeal until they are dependent on the problem's original limitations or predetermined priorities; in scheduling issues, solutions are often produced through operations. These algorithms may "construct" individual processes using "Dispatching Rules," such as programming to discover

a workable solution within the limits of a priority hierarchy. Then, solutions are generated rapidly while preserving their quality. While with local search techniques, the originally created keys are gradually replaced by characteristics learned from a set of surrounding solutions, regardless of whether they originate with a random collection of initial solutions or employ building algorithms [23]. These approaches allow for a more efficient analysis of neighboring solutions in a problem space. These algorithms have the problem of being unable to locate and implement global solutions. Therefore, they can become stuck in the local optimal zone.

Meta-heuristics integrates the heuristic approaches often employed to solve combinatorial optimization problems. Meta-heuristic algorithms use creative search tactics to unearth the global optimum and avoid being entangled in local optima by guiding solution seeking with a bias to acquire more feasible options more rapidly. Some bias mechanisms encompass objective function bias, prejudice based on past judgments, experience bias, etc. [21]. This study employs varied and increased search strategies. Using a metaheuristic technique, the main purpose of the diversification strategy is to efficiently explore all probable solution space neighborhoods. In contrast, the intensification strategy entails utilizing previously acquired search skills and investigating a more localized solution subspace. There are two types of meta-heuristic algorithms: single-point and population-based search. Nature-inspired optimization algorithms are prevalent among population-based algorithms in terms of simplicity of development and greater searchability [24]. For example, the GA (genetic algorithm; affected by evolution) [25, 26], PSO (particle swarm optimization; influenced by swarm intelligence) [27], and SA (simulated annealing; influenced by metal cooling behavior) [28] are among the most dependable and efficient algorithms available.

The coral reef optimization approach (CRO) is one of the complicated bio-inspired computer techniques utilized to address engineering and scientific issues by modeling the "formation" and "reproduction" of corals in coral reefs. In 2014, Salcedo-Sanz et al. were the first to propose the method [14]. Since then, it has been applied to a number of relevant problems, such as optimum mobile network deployment [15], improved battery scheduling of microgrids [16], and wind speed prediction systems with success in "Offshore Wind Farm Design" [17]. Diverse hybrid algorithms adapted from the original version have emerged to improve performance and reduce processing time. In 2016, for instance, a combination of CRO and variable neighborhood search approach was used to solve facility layout difficulties involving uneven area sizes [29]. Alternately, another hybrid CRO approach utilizes Spark's MapReduce programming model to lower the system's total reaction time as well as various other exciting uses [30].

3 Materials and Methods

To accurately represent the problem's reality and improve the efficiency of encoding and decoding, it is vital to pick the most appropriate approach. This decision has a significant effect on the success or failure of problem-solving.

3.1 Representation of JSSP for CRO

The operation solutions are represented as sequences of decimal numbers when the coral reef optimization (CRO) method is utilized to solve JSSP problems. Several alternative techniques, such as operation-based modeling, rule-based priority recognition, machine-based representation, etc., explain the resolution of JSSP depending on the problem's particular features [31, 32]. The two primary subdivisions of these representations are direct and indirect encoding approaches.

Implementing "random keys" is our approach to defining the solution. This method has the benefit of presenting a comprehensive description of the situation. Each number in the sequence represents the number of individual tasks, and the number of repetitions of each location in the sequence shows the number of machines the work must traverse prior to completion. In this study, a "random key" approach reveals a solution to a problem that meets the following criteria:

- Each element in a solution indicates a job to be processed.
- The number of jobs appearing corresponds to the number of machines they must pass through;
- The sequence of the elements in the solution corresponds to the machine order that the job should pass through.

Figure 1 illustrates a "random key" in the context of two computers and three jobs:

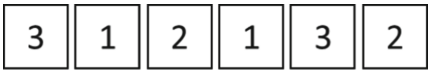

Fig. 1. A random key approach depicts an example JSSP solution.

The CRO algorithm is comprised of two distinct phases: "reef construction" and "coral re-production":

A "reef" is initially created from a M × N square grid. Individual corals are drawn from the population and then randomly distributed in any available vacant square on the reef, following the free/occupation proportion r0 (zero indicating no occupancy). Each coral represents a unique solution in the solution space and will be assigned a health function; the larger the health function, the more likely the corals will survive the algorithm's subsequent generations. The fitness function, which relies on objective functions, calculates the coral health value.

During the second stage, the CRO reproduces coral by repeating five primary mechanisms to build a new coral generation (called larvae): "Broadcast Spawning" (External sexual), "Brooding" (Internal sexual), "Larvae setting," "Budding" (Asexual), and "Depredation."

In Algorithm 1, the operation of the CRO algorithm is given:

Algorithm 1: Coral Reef Optimization (CRO).

Input: $M \times N$: reef size, ρ_0: occupation rate, F_B: fraction of broadcast spawners, F_A: fraction of asexual reproduction, F_D: fraction of the worse fitness corals, P_D: the deprecated probability of the worse fitness corals.

Output: reasonable solution with best fitness

#Initialization—Reef formation phase:

1. $M \times N \leftarrow$ reef size
2. **Generate** initial coral population
3. **Calculate** the fitness value of each coral
4. **Deploy** randomly on the reef with occupied rate ρ_0
5. *#Main loop—Coral reproduction phase:*
6. **Repeat**
7. **Reproduce** coral fraction F_B by **external sexual broadcast spawning**
8. **Reproduce** coral fraction $1 - F_B$ by **internal sexual brooding**
9. **Larvae setting**
10. **Reproduce** best corals fraction F_A by **asexual budding**
11. **Predation** of F_D worst reef corals with P_D probability
12. **Until** *stop_condition*
13. **Return** *best_resonable_solution*

3.2 Objective Function

We utilize "minimize the makespan" as the objective function for solving JSSP in this work. This is the duration between beginning the first job and finishing the last. For a fundamental JSSP with *n* jobs and *m* machines, we have: *Oij* is the operation of the *j*th job conducted on the *i*th computer; *pij* identifies the processing time of the *j*th job done on the *i*th machine based on the beginning time (*rij*) of operation *Oij*; the time required to finish operation *Oij* may then be computed as follows:

$$C_{ij} = r_{ij} + p_{ij} \tag{1}$$

Because machines and jobs have unique and distinct completion times, c_{in} and c_{jm} are defined as the finishing time of the last (*n*th) operation upon *i*th machine and the finishing time of the last (*m*th) operation upon that *j*th job, respectively. Calculating the starting time r_{ij} is as follows:

$$r_{ij} = max(c_{in}, c_{jm}) \tag{2}$$

Lastly, makespan can be computed as the time required to conduct the final operation on the final machine:

$$makespan = C_{max} = max(C_{im}) \tag{3}$$

The efficiency of the scheduling may be determined by comparing the machine's total idle to the whole processing time of the system:

$$C' = 1 + \frac{\sum_i l_i}{\sum_{j,k} p_{jk}} = \frac{C \cdot m}{\sum_{j,k} p_{jk}} \tag{4}$$

where li denotes the idle time of machine i; m denotes the number of machines; p_{jk} represents the processing time of job i on machine k; C means makespan;

When applied to the JSSP, the method evaluates solution quality using the Objective and Fitness functions obtained from Eqs. (5) and (6), respectively. The Fitness function drives the optimization process by expressing how the suggested solution matches the given objective inextricably. In contrast, the Objective function reveals how "great" the solution is in terms of the optimized function's performance.

$$f = \frac{1}{C_{max}} \tag{5}$$

$$F_{(i)} = \frac{f_{(i)}}{\sum_1^n f_{(i)}} \tag{6}$$

where C_{max} (or makespan) is the period of time between commencing a job and finishing the last one, n is the population size and $f_{(i)}$ is the fitness function.

3.3 Local Search: Simulated Annealing (SA)

SA is the applied local search algorithm for CRO [33]. It is a technique used to imitate the cooling behavior of metal exposed to high temperatures. The metal is quickly heated up to a high temperature and then gently cooled in accordance with a "cooling schedule" in order to create the optimal crystalline structure with the lowest possible internal energy. High temperature imparts crystal grains with a high degree of energy, allowing them to "jump" easily and swiftly to their designated positions within the crystal structure. During the cooling process, the temperature gradually declines, and crystals are expected to be at their ideal positions [28] after the temperature has been sufficiently lowered. Algorithm 2 describes the SA as follows:

Algorithm 2: Simulated Annealing

Input: t: temperature, t_{min}: min temperature, α: cooling rate, F: fitness function, S: solution, $maxIter$: maximum iteration
Output: Best_solution
#Initialization:
1. $t \leftarrow$ initial temperature
2. Best_solution $\leftarrow S$
#Main loop:
While $t > t_{min}$ **do**
3. iter $\leftarrow 0$
4. **While** iter $<$ $maxIter$ **do**
5. **Select** a random solution S'
6. $\Delta \leftarrow F(S') - F(S)$
7. **If** $\Delta < 0$ do
8. S $\leftarrow$S'
9. **If** $F(S') < F$(Best_solution) **do**
10. Best_solution $\leftarrow$S'
11. **Else if** $rand(0,1) < e^{-\Delta/t}$ **do**
12. S $\leftarrow$S'
13. iter $\leftarrow$ iter $+ 1$
14. t $\leftarrow$ t $* \alpha$
15. **Return**

3.4 Proposal Approaches

We propose a hybrid algorithm called CROLS based on the CRO depicted in Fig. 3's flowchart.:

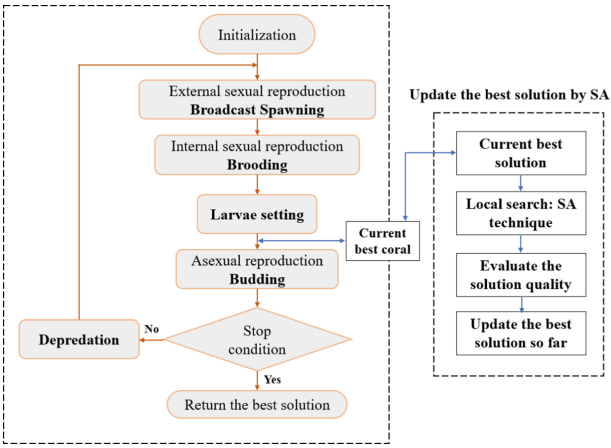

Fig. 3. CRO applied local search techniques.

Typically, algorithms that determine the best answer from a random starting solution, such as the CRO, require considerable time to provide an ideal result. On occasion, the

search process becomes stuck in the local optimum and is unable to converge to the optimal solution. Combining local search methodologies is advised to shorten convergence time towards the optimal solution, prevent unnecessary local optimal, and increase search efficiency. Algorithm 3 describes the CROLS algorithm as follows:

Algorithm 3: Hybrid Coral Reef Optimization Algorithms (CROLS)

Input: $M \times N$: reef size, ρ_0: occupation rate, F_B: fraction of broadcast spawners, F_A: fraction of asexual reproduction, F_D: fraction of the worse fitness corals, P_D: the deprecated probability of the worse fitness corals.

Output: Reasonable solution with best fitness

#Initialization—Reef formation phase:

1. $M \times N \leftarrow$ reef size
2. **Generate** initial coral population
3. **Calculate** the fitness value of each coral
4. **Deploy** randomly on the reef with occupied rate ρ_0
5. *#Main loop—Coral reproduction phase:*
6. **Repeat**
7. **Reproduce** coral fraction F_B by **external sexual broadcast spawning**
8. **Reproduce** coral fraction $1 - F_B$ by **internal sexual brooding**
9. **Larvae evaluation**
10. **Larvae setting**
11. **Apply** *"Local search strategy:* SA*"*
12. **Reproduce** best corals fraction F_A by **asexual budding**
13. **Predation** of F_D worst reef corals with P_D probability
14. **Until** *stop_condition*
15. **Return** *best _solution*

The decision variables of the problem were handled by P_D probability in the proposed algorithm.

4 Experiment Results and Discussion

This section presents a concise and accurate summary of the experimental findings, their interpretation, and the investigation's conclusion.

4.1 Parameters Used in the Algorithm

A number of metaheuristic optimization algorithms aim their search for the best solution with parameters created at random. Therefore, creating the criteria is one of the most crucial steps. Based on the theory of basic evolutionary algorithms and JSSP tests, we determined optimal parameters for the hybrid algorithm. The following parameters are listed in Table 1:

Based on the experimental results, we think that the algorithm's parameters may vary depending on the problem's complexity and size. Even if the setup factors are equal, the outcomes in various situations are not identical. To find the optimal solution for issues of

Table 1. Parameters used in the hybrid algorithms.

Parameter	Technique	Definition	Range
$M \times N$	CRO	Reef size	$[10 \times 10, 30 \times 30]$
Iter	CRO	Number of iterations (generations)	[50, 200]
F_b	CRO	Probability of Broadcast spawning process	[0.8, 0.9]
F_a	CRO	Probability of Budding process	[0.05, 0.15]
r_0	CRO	Initial free/occupied ratio	[0.6, 0.8]
F_d	CRO	Probability of selecting weak individuals from the population	[0.01, 0.1]
P_d	CRO	Probability of removing weak individuals from the population	[0.01, 0.1]
k	CRO	Number of chances for a new coral to colonize a reef	[2, 4]
ke	CRO	Maximum number of allowed equal corals	[0.1, 0.3]
t_{min}	SA	Min temperature	[0.0001, 1]
α	SA	Cooling rate	[0.7, 0.99]

diverse sizes in an acceptable amount of time, it is required to experiment with different configuration choices. Using a series of tests, we employed the trial-and-error approach for adjusting parameters in order to obtain an acceptable parameter set. Table 2 provides a summary of recommended parameter settings for CROLS with distinct issue sizes.

Table 2. Suggested parameters set for different sizes of problem.

Size	$M \times N$	r_0	Fb	Fa	Fd	Pd	k	$tmin$	α
Small	10×10	0.6	0.9	0.05	0.01	0.1	3	0.5	0.85
Medium	20×20	0.7	0.85	0.05	0.05	0.1	3	0.5	0.85
Large	30×30	0.7	0.85	0.1	0.1	0.1	3	0.5	0.85

4.2 Experiment Results

The experimental strategy is divided into two steps: first, we will assess the influence of local search on CRO using an aggregated CRO and hybrid algorithm results in Table 3. Second, we compare the best search results of CROLS with five state-of-the-art algorithms from the literature to demonstrate that the proposed hybrid algorithms have advantageous search capabilities. The suggested method is written in Python on a computer with an Intel (R) Core i5 3.1 GHz CPU and 24 GB RAM.

4.2.1 Search Performance on CRO-Based Algorithm with Different Reef Sizes

We will assess the influence of local search on CRO using an aggregated CRO and hybrid algorithm outcome table. In this experiment, we utilize a subset of JSSP by Lawrence (1984) (LA) [34], one of the common cases for addressing JSSP, to assess the efficiency of the method with a population size of 30 × 30 and all parameters set by Table 2. The worst, mean, best and standard deviation (SD) of the makespan score determined from 30 separate executions of each method on 16 JSSP instances are shown in Table 3.

Table 3. Statistics for 16 LA cases using CRO-based methods.

Instance	Size	Method	Opt	Best	Worst	Mean	SD	Time (s)
LA01	10 × 5	CRO	666	666	674	669.55	2.7	44.27
		CROLS	666	666	671	668.09	2	20.27
LA02	10 × 5	CRO	655	655	676	665.36	6.62	44.55
		CROLS	655	655	671	663.18	6.02	18.55
LA06	15 × 5	CRO	926	926	946	935	7.82	76.36
		CROLS	926	926	940	931.18	5.04	46.55
LA07	15 × 5	CRO	890	890	922	908.18	7.38	78.09
		CROLS	890	890	916	906.91	5.29	45.91
LA11	20 × 5	CRO	1222	1222	1257	1233.45	12.58	128.82
		CROLS	1222	1222	1244	1230.73	8.52	100.18
LA12	20 × 5	CRO	1039	1039	1060	1048.27	7.96	120.09
		CROLS	1039	1039	1043	1047.45	7.27	103.45
LA16	10 × 10	CRO	945	948	1011	976.55	19.06	177.64
		CROLS	945	945	955	947.73	2.93	125.91
LA17	10 × 10	CRO	784	788	832	808.27	13.51	183.18
		CROLS	784	784	799	788.55	4.88	128.82
LA21	15 × 10	CRO	1046	1106	1250	1172.64	51.47	256.64
		CROLS	1046	1046	1066	1052.55	7.76	164.36
LA22	15 × 10	CRO	927	935	1150	1030.09	53.54	248.27
		CROLS	927	927	944	932.73	5.69	185.18
LA26	20 × 10	CRO	1218	1226	1275	1255.82	44.37	359.91
		CROLS	1218	1218	1246	1229.91	10.75	281.73
LA27	20 × 10	CRO	1235	1255	1375	1359.55	25.96	336.18
		CROLS	1235	1235	1246	1239.09	4.2	260.18
LA32	30 × 10	CRO	1850	1912	2105	1998.52	30.26	566.64
		CROLS	1850	1850	1856	1853	2.05	453.45

(*continued*)

Table 3. (*continued*)

Instance	Size	Method	Opt	Best	Worst	Mean	SD	Time (s)
LA33	30 × 10	CRO	1719	1805	2034	1934.91	60.59	557.82
		CROLS	1719	1719	1732	1723.45	4.97	441.73
LA39	15 × 15	CRO	1233	1332	1403	1397.21	24.37	752.09
		CROLS	1233	1233	1255	1243.09	34.44	592.64
LA40	15 × 15	CRO	1222	1342	1424	1374.33	24.88	739.82
		CROLS	1222	1228	1264	1238.25	15.89	585.45

Table 3 shows that the local search strategy has made the CRO algorithm more stable, as the amplitude of the mean between the best to the worst and the standard deviation in all cases have improved. CROLS also significantly reduces the search duration compared to the original method. Technically, CROLS has 20% more fitness function calls than the original algorithm since they employ local search techniques. Even though this increases the amount of computational work, it helps the algorithm converge quickly and experimentally shows that CROLS is still more rapid at search-than the original CRO.

To evaluate the improvement of the CROLS to CRO statistically, we performed Friedman's test on the data in Table 3. The "mean rank" and Friedman's test statistic results of three algorithms participating in the evaluation will be described in Tables 4 and 5, respectively.

Table 4. Mean rank of two CRO-based algorithms.

Algorithm	Mean Rank
CRO	2.94
CROLS	1.47

Table 5. Friedman's test statistic of two CRO-based algorithm.

	X^2	ρ
CRO-CROLS	21.556	0.000021

According to Table 4, there is little difference in the mean rank of the CRO algorithm between reef sizes. The mean rank of CROLS is significantly different from CRO, indicating that local search techniques substantially impact these two algo-rithms. Table 5 provides the test statistics for Friedman's test with a significance level $\alpha = 0.05$. All χ^2 values are larger than the critical value $\chi_0^2 = 5.991$, and all ρ values are less than 0.05,

demonstrating statistically significant differences in the per-formance of two CRO-based algorithms.

Following Friedman's test made a significant result, we performed Wilcoxon's signed-rank test as a posthoc analysis to determine the difference between three CRO-based algorithms in pairwise group. The search findings of three reef sizes will be averaged to feed the Wilcoxon's test with a statistical significance of 0.05. Table 6 provides the statistical results of the Wilcoxon's test.

Table 6. Wilcoxon's test statistic of two CRO-based algorithms.

	CRO-CROLS
Z	−3.516
p	0.000438

Using the Z distribution table, we can find the critical value of Z is 1.96 at a statistical significance of 0.05. According to Table 6, the statistical findings of the pair CRO-CROLS is $Z = -3.516$ ($|Z| > 1.96$) and $p = 0.000438 < 0.05$. So, the null hy-pothesis is rejected at statistical significance of 0.05; the advantage of local search approaches on CROLS compared to the original CRO algorithm is statistically significant.

4.3 Improvement of Search Efficiency

To measure the performance increase of the two hybrid algorithms, we use the mean deviation from the optimum makespan of CROLS that outperforms the original technique CRO in terms of search performance. The following formula is used to determine the evolution of search effectiveness:

$$\boldsymbol{PI}_{(\boldsymbol{CROLS/CRO})} = \frac{\boldsymbol{S}_{(\boldsymbol{CRO})} - \boldsymbol{S}_{(\boldsymbol{CROLS})}}{\boldsymbol{S}_{(\boldsymbol{Opt})}} \times 100\% \tag{7}$$

where $\boldsymbol{PI}_{(\boldsymbol{CROLS/CRO})}$ is the percentage of improvement, $\boldsymbol{S}_{(\boldsymbol{CRO})}$ is the minuscule makespan by CRO, $\boldsymbol{S}_{(\boldsymbol{CROLS})}$ is the minuscule makespan by CROLS, $\boldsymbol{S}_{(\boldsymbol{Opt})}$ is the minuscule makespan in comparing between CRO and CROLS.

Table 7 depicts the percentage performance improvement of CROLS over CRO.

Analyzing Table 7 demonstrates that the local search techniques have improved the search performance of the original CRO algorithm. Most PI measurements are below 3% in the basic situations (LA01–LA22), indicating that the improvement is not all that evident. However, as the issue complexity rose, CROLS performed far better than the original CRO. Even in LA40, where it is difficult for the CRO to identify the best-known solution, the PI achieves a high of roughly 10%, which happens in reefs of all sizes. This highlights the effect that local search algorithms have on search performance.

Table 7. Improve performance of hybrid algorithm.

Instance	Size	PI
LA01	10 × 5	0
LA02	10 × 5	0.15
LA06	15 × 5	0
LA07	15 × 5	0
LA11	20 × 5	0
LA12	20 × 5	0
LA16	10 × 10	0.32
LA17	10 × 10	0.51
LA21	15 × 10	0.96
LA22	15 × 10	0.86
LA26	20 × 10	0.66
LA27	20 × 10	1.62
LA32	30 × 10	3.35
LA33	30 × 10	5
LA39	15 × 15	8.03
LA40	15 × 15	9.33

4.3.1 Comparing the Computational Outcome of CRO-Based Algorithms to Other State-of-the-Art Algorithms

To further evaluate the efficacy of CRO and suggested hybrid algorithms, we conducted comprehensive tests on the 12 previously described LA instances, Fisher and Thompson [35]: FT06, FT10, FT20; Applegate and Cook [36]: ORB01–ORB09; and five of Adams et al. [37] marked ABZ05 to ABZ09. The results of two CRO-based algorithms were compared to those of five state-of-the-art algorithms discovered in the literature: hybrid PSO enhanced with nonlinear inertia weight, and Gaussian mutation (NGPSO) [38], multi-Crossover Local Search Genetic Algorithm (mXLSGA) [39], single seekers society (SSS) algorithm [40], genetic algorithm with a critical-path-guided Giffler and Thompson crossover operator (GA-CPG-GT). As a performance criteria, the best makespan produced by the CRO-based algorithms with reef size 30 × 30 from 30 separate runs was employed. Table 8 displays the experimental findings for the 34 cases, including the instance name, problem size (number of jobs × number of machines), best-known solution (BKS), and best solution obtained by each comparison method.

We recognize that the original CRO algorithm can only find the best value in a few simple scenarios, such as LA01; LA02; LA06; LA07; LA11; LA12 and FT06, but the rest of the CRO results are acceptable and are comparable to GA-CPG-GT. While the CROLS showed efficiency, it also produced outcomes that were superior or on par with those of five cutting-edge algorithms in 11/12 LA instances, 2/3 FT instances, 3/5 ABZ

Table 8. Experimental findings for 34 instances are compared with CRO-based algorithms and other state-of-the-art algorithms. "Not evaluated in that instance" is denoted by the symbol "-."

Instance	Size	BKS	CRO	CROLS	mXLSGA (2020)	NGPSO (2020)	SSS (2020)	GA-CPG-GT (2019)	DWPA (2019)
LA01	10 × 5	666	**666**	**666**	**666**	**666**	**666**	**666**	**666**
LA02	10 × 5	655	**655**	**655**	**655**	**655**	**655**	**655**	**655**
LA06	15 × 5	926	**926**	**926**	**926**	**926**	**926**	**926**	**926**
LA07	15 × 5	890	**890**	**890**	**890**	**890**	**890**	**890**	**890**
LA11	20 × 5	1222	**1222**	**1222**	**1222**	**1222**	**1222**	**1222**	**1222**
LA12	20 × 5	1039	**1039**	**1039**	**1039**	**1039**	**-**	**1039**	**1039**
LA16	10 × 10	945	955	**945**	**945**	**945**	947	946	993
LA17	10 × 10	784	788	**784**	**784**	794	-	**784**	793
LA21	15 × 10	1046	1056	**1046**	1059	1183	1076	1090	1105
LA22	15 × 10	927	935	**927**	935	**927**	-	954	989
LA26	20 × 10	1218	1226	**1218**	**1218**	**1218**	-	1237	1303
LA27	20 × 10	1235	1255	**1235**	1269	1394	-	1313	1346
LA32	30 × 10	1850	1912	**1850**	**1850**	**1850**	**-**	**1850**	**1850**
LA33	30 × 10	1719	1805	**1719**	**1719**	**1719**	**-**	**1719**	**1719**
LA39	15 × 15	1233	1332	**1233**	1258	1662	-	1290	1334
LA40	15 × 15	1222	1342	1228	1243	**1222**	1252	1252	1347
FT06	6 × 6	55	**55**	**55**	**55**	**55**	**55**	**55**	-

(*continued*)

Table 8. (*continued*)

Instance	Size	BKS	CRO	CROLS	mXLSGA (2020)	NGPSO (2020)	SSS (2020)	GA-CPG-GT (2019)	DWPA (2019)
FT10	10 × 10	930	934	**930**	**930**	**930**	936	935	-
FT20	20 × 5	1165	1197	1174	**1165**	1210	**1165**	1180	-
ABZ05	10 × 10	1234	1255	**1234**	**1234**	**1234**	-	1238	-
ABZ06	10 × 10	943	988	**943**	**943**	**943**	-	947	-
ABZ07	20 × 15	656	755	731	**695**	713	-	-	-
ABZ08	20 × 15	665	720	709	713	729	-	-	-
ABZ09	20 × 15	679	817	**707**	721	930	-	-	-
ORB01	10 × 10	1059	1120	1070	**1068**	1174	-	1084	-
ORB02	10 × 10	888	927	899	**889**	913	-	890	-
ORB03	10 × 10	1005	1097	**1021**	1023	1104	-	1037	-
ORB04	10 × 10	1005	1121	**1005**	**1005**	**1005**	-	1028	-
ORB05	10 × 10	887	904	890	889	**887**	-	894	-
ORB06	10 × 10	1010	1085	1020	**1019**	1124	-	1035	-
ORB07	10 × 10	397	418	**397**	**397**	**397**	-	404	-
ORB08	10 × 10	899	988	912	**907**	1020	-	937	-
ORB09	10 × 10	934	955	**938**	940	980	-	943	-
ORB10	10 × 10	944	1010	967	**944**	1027	-	967	-

cases, and 5/10 ORB occurrences. The results of CROLS surpass SSS, GA-CPG-GT, DWPA algorithms, as well as competitive comparisons with mXLSGA and NGPSO. In

certain cases, such as ORB01; ORB03; ORB08; and ORB10; CROLS even outperforms NGPSO.

5 Conclusions

This study introduces the innovative hybrid algorithm CROLS, which makes use of many search techniques. In order to locate the global optimal solution and avoid the local optimal, the CROLS method uses simulated annealing (SA). This method reduces execution time while increasing the chance of getting the best result. This search strategy uses the potential of the entire reef and increases its convergence in an effort to produce new, superior individuals. The article provides insights into how the local search approach to the CRO algorithm might be improved, as shown by encouraging testing results. When the hybrid algorithm's best search results are compared to the best-known outcomes, its effectiveness is further demonstrated. For more complex problems, a variety of multi-objective optimization methods like JSSP have been proposed. In the future plan, we ought to create the fastest processing algorithm possible for multi-objectives problems. An attractive future research direction to improve the applicability of JSSP in manufacturing is the development of algorithms to handle multi-objective optimization problems.

Acknowledgment. This research was partly supported by National Science and Technology Council, Taiwan with grant numbers 111-2221-E-992-066 and 109-2221-E-992-073-MY3.

References

1. Marco Baptista: How important is production scheduling today? Opcenter (2020)
2. Ben Hmida, J., Lee, J., Wang, X., Boukadi, F.: Production scheduling for continuous manufacturing systems with quality constraints. Prod. Manuf. Res. **2**, 95–111 (2014)
3. Jiang, Z., Yuan, S., Ma, J., Wang, Q.: The evolution of production scheduling from Industry 3.0 through Industry 4.0. Int. J. Prod. Re. **60**, 3534–3554 (2022)
4. Graves, S.C.: A review of production scheduling. Oper. Res. **29**(4), 646–675 (1981)
5. Johnson, S.M.: Optimal two- and three-stage production schedules with setup times included. Naval Res. Logistics Quart. **1**, 61–68 (1954)
6. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Math. Oper. Res. **1**, 117–129 (1976)
7. Zhang, J., Ding, G., Zou, Y., Qin, S., Fu, J.: Review of job shop scheduling research and its new perspectives under Industry 4.0. J. Intell. Manuf. **30**(4), 1809–1830 (2017). https://doi.org/10.1007/s10845-017-1350-2
8. Xiong, H., Shi, S., Ren, D., Hu, J.: A survey of job shop scheduling problem: the types and models. Comput. Oper. Res. **142**, 105731 (2022)
9. Xhafa, F., Abraham, A.: Metaheuristics for Scheduling in Industrial and Manufacturing Applications (2022)
10. Pinedo, M.L.: Planning and Scheduling in Manufacturing and Services (2022)
11. Türkyılmaz, A., Şenvar, Ö., Ünal, İ, Bulkan, S.: A research survey: heuristic approaches for solving multi objective flexible job shop problems. J. Intell. Manuf. **31**(8), 1949–1983 (2020). https://doi.org/10.1007/s10845-020-01547-4

12. Guzman, E., Andres, B., Poler, R.: Matheuristic algorithm for job-shop scheduling problem using a disjunctive mathematical model. Computers **11**, 1 (2022)
13. Viana, M.S., Contreras, R.C., Morandin Junior, O.: A New frequency analysis operator for population improvement in genetic algorithms to solve the job shop scheduling problem. Sensors **22**, 4561 (2022)
14. Salcedo-Sanz, S., Del Ser, J., Landa-Torres, I., Gil-López, S., Portilla-Figueras, J.A.: The coral reefs optimization algorithm: a novel metaheuristic for efficiently solving optimization problems. Sci. World J. **2014**, e739768 (2014)
15. Salcedo-Sanz, S., García-Díaz, P., Portilla-Figueras, J.A., Del Ser, J., Gil-López, S.: A coral reefs optimization algorithm for optimal mobile network deployment with electromagnetic pollution control criterion. Appl. Soft Comput. **24**, 239–248 (2014)
16. Salcedo-Sanz, S., Camacho-Gómez, C., Mallol-Poyato, R., Jiménez-Fernández, S., Del Ser, J.: A novel coral reefs optimization algorithm with substrate layers for optimal battery scheduling optimization in micro-grids. Soft. Comput. **20**(11), 4287–4300 (2016). https://doi.org/10.1007/s00500-016-2295-7
17. Salcedo-Sanz, S., et al.: Offshore wind farm design with the coral reefs optimization algorithm. Renew. Energy **63**, 109–115 (2014)
18. Bedoya-Valencia, L.: Exact and Heuristic Algorithms for the Job Shop Scheduling Problem with Earliness and Tardiness over a Common Due Date. Old Dominion University (2007)
19. Brucker, P., Jurisch, B., Sievers, B.: A branch and bound algorithm for the job-shop scheduling problem. Discret. Appl. Math. **49**, 107–127 (1994)
20. Çaliş, B., Bulkan, S.: A research survey: review of AI solution strategies of job shop scheduling problem. J. Intell. Manuf. **26**(5), 961–973 (2013). https://doi.org/10.1007/s10845-013-0837-8
21. Muthuraman, S., Venkatesan, V.P.A.: Comprehensive study on hybrid meta-heuristic approaches used for solving combinatorial optimization problems. In: 2017 World Congress on Computing and Communication Technologies (WCCCT), pp. 185–190 (2017). https://doi.org/10.1109/WCCCT.2016.53
22. Aarts, E., Lenstra, J.K. (eds.): Local Search in Combinatorial Optimization. Princeton University Press (2003)
23. Gendreau, M., Potvin, J.-Y.: Metaheuristics in combinatorial optimization. Ann Oper Res **140**, 189–213 (2005)
24. Yang, X.-S. (ed.): Nature-Inspired Optimization Algorithms. Elsevier (2014)
25. Davis, L.: Job shop scheduling with genetic algorithms. In: Proceedings of the 1st International Conference on Genetic Algorithms, pp. 136–140. L. Erlbaum Associates Inc. (1985)
26. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press eBooks, IEEE Xplore (2022)
27. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 – International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
28. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
29. Garcia-Hernandez, L., Salas-Morera, L., Carmona-Muñoz, C., Abraham, A., Salcedo-Sanz, S.: A hybrid coral reefs optimization—variable neighborhood search approach for the unequal area facility layout problem. IEEE Access **8**, 134042–134050 (2020)
30. Tsai, C.-W., Chang, H.-C., Hu, K.-C., Chiang, M.-C.: Parallel coral reef algorithm for solving JSP on Spark. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (2016)
31. Cheng, R., Gen, M., Tsujimura, Y.: A tutorial survey of job-shop scheduling problems using genetic algorithms—I. representation. Comput. Indus. Eng. **30**(4), 983–997 (1996)

32. Cheng, R., Gen, M., Tsujimura, Y.: A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies. Comput. Ind. Eng. **36**, 343–364 (1999)
33. Lee, Y.S., Graham, E., Jackson, G., Galindo, A., Adjiman, C.S.: A comparison of the performance of multi-objective optimization methodologies for solvent design. In: Kiss, A.A., Zondervan, E., Lakerveld, R., Özkan, L. (eds.) Computer Aided Chemical Engineering, vol. 46, pp. 37–42. Elsevier (2019)
34. Lawrence, S.: Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement). Graduate School of Industrial Administration. Pittsburgh, Pennsylvania, Carnegie-Mellon University (1984)
35. Fisher, C., Thompson, G.: Probabilistic Learning Combinations of Local Job-shop Scheduling Rules, pp. 225–251. Industrial Scheduling (1963)
36. Applegate, D., Cook, W.: A computational study of the job-shop scheduling problem. ORSA J. Comput. **3**, 149–156 (1991)
37. Adams, J., Balas, E., Zawack, D.: The shifting bottleneck procedure for job shop scheduling. Manage. Sci. **34**, 391–401 (1988)
38. Yu, H., Gao, Y., Wang, L., Meng, J.: A hybrid particle swarm optimization algorithm enhanced with nonlinear inertial weight and gaussian mutation for job shop scheduling problems. Mathematics **8**, 1355 (2020)
39. Viana, M.S., Junior, O.M., Contreras, R.C.: An improved local search genetic algorithm with multi-crossover for job shop scheduling problem. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) ICAISC 2020. LNCS (LNAI), vol. 12415, pp. 464–479. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61401-0_43
40. Hamzadayı, A., Baykasoğlu, A., Akpınar, Ş: Solving combinatorial optimization problems with single seekers society algorithm. Knowl.-Based Syst. **201–202**, 106036 (2020)