Computers & Industrial Engineering 88 (2015) 273–283

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A new hybrid island model genetic algorithm for job shop scheduling problem ☆

Mohamed Kurdi *

Computer Engineering Department, Adnan Menderes University, 09010 Aydin, Turkey

ARTICLE INFO

Article history:
Received 26 January 2015
Received in revised form 12 July 2015
Accepted 13 July 2015
Available online 30 July 2015

Keywords:
Job shop scheduling
Island model genetic algorithm
Tabu search
Parallel hybrid metaheuristics

ABSTRACT

This paper presents a new hybrid island model genetic algorithm (HIMGA) to solve the well-known job shop scheduling problem (JSSP) with the objective of makespan minimization. To improve the effectiveness of the island model genetic algorithm (IMGA), we have proposed a new naturally inspired self-adaptation phase strategy that is capable of striking a better balance between diversification and intensification of the search process. In the proposed self-adaptation phase strategy, the best individuals are recruited to perform a local search using tabu search (TS), and the worst ones are recruited to perform a global search using a combination of 3 classical random mutation operators. The proposed algorithm is tested on 76 benchmark instances, with the proposed self-adaptation strategy, and without it using the classical alternatives, and also compared with other 15 algorithms recently reported in the literature. Computational results verify the improvements achieved by the proposed self-adaptation strategy, and show the superiority of the proposed algorithm over 13 of the compared works in terms of solution quality, and validate its effectiveness.

1. Introduction

Job shop scheduling problem (JSSP) is an NP-hard problem, and one of the most intractable combinatorial optimization problems considered to date. This intractability and importance for industrial engineering in terms of improving machine utilization and reducing cycle-time, made it so widely studied for more than fifty years.

Earlier works on JSSP were centered on exact methods such as branch and bound to find optimal solutions for small size problems (Carlier & Pinson, 1989; Lageweg, Lenstra, & Rinnooy Kan, 1977); however, they failed in solving problems of a bigger size in practical computational cost. For that reason, research focus was shifted towards approximation methods, which do not guarantee finding optimal solutions, but there is a considerable probability of finding near optimal solutions in practical computational cost. Initially, they were limited to simple heuristic methods such as dispatching rules (Blackstone, Phillips, & Hogg, 1982) and shifting bottleneck procedure (Adams, Balas, & Zawack, 1988) which were distinguished in terms of efficiency, but undistinguished in terms of effectiveness. Therefore, research focus was shifted again towards more sophisticated approximation methods called metaheuristics, which explore the search space more effectively by employing more intelligence in escaping from local optimums. The most common ones that have been used for JSSP include genetic algorithm (GA) (Watanabe, Ida, & Gen, 2005), simulated annealing (SA) (Satake, Morikawa, Takahashi, & Nakamura, 1999; Van Laarhoven, Aarts, & Lenstra, 1992), tabu search (TS) (Nowicki & Smutnicki, 1996; Zhang, Li, Guan, & Rao, 2007), ant colony optimization (ACO) (Fığlalı, Özkale, Engin, & Fığlalı, 2009), and particle swarm optimization (PSO) (Lian, Jiao, & Gu, 2006; Lin et al., 2010). However, due to the stubborn nature of JSSP, sole metaheuristic methods left a considerable space of improvements; consequently, recently, most of researchers tend to develop hybrid methods that combine the complementary strengths of different metaheuristic. Actually, to the best of our knowledge, the best-so-far method for JSSP is a tabu search/path relinking method proposed by Peng, Lü, and Cheng (2015). An overview of JSSP techniques can be found in Zobolas, Tarantilis, and Ioannou (2008), while a comprehensive survey of them can be found in Jain and Meeran (1999).

GA has gained a well-earned reputation in being one of the best methods in solving JSSP, but it still has its own shortcomings like premature convergence, and the lack of intensification capabilities, i.e. searching in the small regions of the search space that are likely close to the optimal solutions (Zobolas et al., 2008). Various approaches have been developed during the last 4 decades to overcome these shortcomings, the most common ones include

☆ This manuscript was processed by Area Editor T.C. Edwin Cheng.

* Tel.: +90 256 213 7503x3577; fax: +90 256 213 6686.
E-mail address: mohamed.kurdi@adu.edu.tr

http://dx.doi.org/10.1016/j.cie.2015.07.015
0360-8352/

parallelization of GA into multiple sub-populations such as the island model GA (IMGA) to delay the premature convergence and improve the search diversification capabilities (Asadzadeh & Zamanifar, 2010; Gu, Gu, & Gu, 2009; Park, Choi, & Kim, 2003; Qi, Burns, & Harrison, 2000; Yusof, Khalid, Hui, Md Yusof, & Othman, 2011), and hybridization with local search methods (LSMs) to add intensification capabilities to it such as TS (Amirghasemi & Zamani, 2015; Cheng, Peng, & Lü, 2013; Meeran & Morshed, 2014; Ombuki & Ventresca, 2004), SA (Tamilarasi & kumar, 2010; Wang & Zheng, 2001), and new LSMs (Asadzadeh, 2015; Gao, Zhang, Zhang, & Li, 2011; Qing-dao-er-ji & Wang, 2012; Zhou, Feng, & Han, 2001). A tutorial survey of JSSP using GAs can be found in Cheng, Gen, and Tsujimura (1996), while a tutorial survey of JSSP using hybrid GAs can be found in Cheng, Gen, and Tsujimura (1999).

Recently, it has been shown that combining both of parallelization and hybridization in one framework is advantageous. Kalantari and SanieeAbadeh (2013) proposed an IMGA hybridized with a LSM based on pair-wise interchanged method, but this kind of LSMs acts like a random swap mutation (Gen & Cheng, 1997), cannot improve the makespan, and may generate infeasible solutions since there is no guarantee that the interchanged operations do belong to the critical path (Taillard, 1994); Asadzadeh (2014) proposed an IMGA hybridized with variable neighborhood search (VNS), but the VNS is applied on each individual after the end of each generation, and involves just two random mutation operators. Therefore, both of them may not achieve a proper balance between diversification and intensification.

It is known that there are two phases of evolution in GA: the cooperation phase implemented by crossover, and self-adaptation phase implemented by mutation. Whereas cooperation phase means individuals evolve by exchanging their information about the search space, self-adaptation phase means individuals evolve independently using only their own information (Hertz & Kobler, 2000).

In this work, in order to overcome the limitations of the previously discussed hybrid IMGA models, and design a new effective algorithm that can strike a better balance between diversification and intensification, an IMGA hybridized with TS (which is one of the best LSMs for JSSP) (HIMGA) is proposed. The proposed algorithm utilizes a new naturally inspired self-adaptation phase strategy, in which the best individuals are recruited to perform a local search using TS, and the worst ones are recruited to perform a global search using a combination of 3 classical random mutation operators.

The remainder of this paper is organized as follows. In the next section, JSSP definition is presented. In Section 3, the HIMGA framework is explained. Section 4 presents the computational results. The conclusions are made in Section 5.

2. Problem definition

The classical JSSP with the objective of makespan minimization, which is represented by $J||C_{\max}$ using the classification scheme of Graham, Lawler, Lenstra, and Kan (1979), consists of a set of n jobs $\{J_j\}1 \leqslant j \leqslant n$ needs to be processed on a set of m machines $\{M_r\}1 \leqslant r \leqslant m$. The processing of job J_j on machine M_r is called the operation O_{jr}, and lasts for an uninterrupted specified time period called processing time P_{jr} (preemption is not allowed). Two constraints are imposed on the problem: the precedence constraint which specifies that each job J_j should be processed on each machine M_r according to a predefined sequence called topological sequence, and the capacity constraint which specifies that each machine M_r can process only one job J_j at a time. The start time and completion time of operation O_{jr} are denoted as S_{jr}, C_{jr} respectively. A schedule (or solution) is the set of completion times for all operations; a feasible schedule is a schedule that satisfies the problem constraints. The time needed for the completion of all the operations is called makespan and denoted as $C_{\max}$, where $C_{\max} = \max_{1 \leqslant j \leqslant n,\ 1 \leqslant r \leqslant m} C_{jr}$. The objective of the problem becomes finding a feasible schedule that minimizes $C_{\max}$ as much as possible.

An example of a 3×3 JSSP is given in Table 1. The data include the topological sequence of all jobs with their processing times, for example, job 2 is processed in this order $O_{21} \rightarrow O_{22} \rightarrow O_{23}$, i.e. it is processed on machine 1 for 4 time units, then on machine 2 for 5 units, then on machine 3 for 3 units. A possible solution of the 3×3 JSSP represented by a Gantt chart is given in Fig. 1.

3. The hybrid island model genetic algorithm

Current implementations of parallel GAs (PGAs) are classified into three models: master–slave, fine-grained, and IMGA. In the first, there is one population, and computation is carried out by many processors; therefore, the quality of solutions is preserved while reducing the computational cost. In the second, there is also one population, but each individual is assigned to one processor to carry out only its operations, selection and reproduction are limited to neighbors. In the third, (which is also called multi-population or coarse-grained) the whole population is split into multiple independent islands (sub-populations) that are assigned to different processors, and explore different parts of the search space using their own evolution processes, with occasionally exchange of information via a migration policy (Engelbrecht, 2007); this imitates the nature in a better way, delays the premature convergence, and enhances the search diversification (Konfrst, 2004).

In this work, we adopt IMGA, because it is easy to implement in a serial manner (pseudo-parallel) on a single processor system, which serves our objectives that are centered on effectiveness rather than efficiency.

For the sake of simplicity, the LSM adopted is the classical TS algorithm, based on the one proposed by Nowicki and Smutnicki (1996), which has been found to be particularly successful approach for the JSSP. However, our method does not integrate its diversification procedure (e.g., long-term memory and the back-jump tracking procedure) which is used to restart the search process when gets trapped in a local optimum, because the proposed algorithm will count on TS for doing only the intensification part and leave the task of diversification to IMGA, i.e. IMGA will identify the promising regions, whose local optimums will be subsequently located by TS. The general framework of the proposed algorithm HIMGA is described in Fig. 2.

3.1. Migration policy

It plays a crucial role in the performance, since it controls the process of information exchange between islands. Its mechanism depends mainly on four factors: communications topology, migration rate, selection method, and replacement method (Engelbrecht, 2007).

Table 1
An example of a 3×3 JSSP.

Job	Machine/processing time		
J1	M1/3	M3/4	M2/9
J2	M1/4	M2/5	M3/3
J3	M2/4	M3/6	M1/4

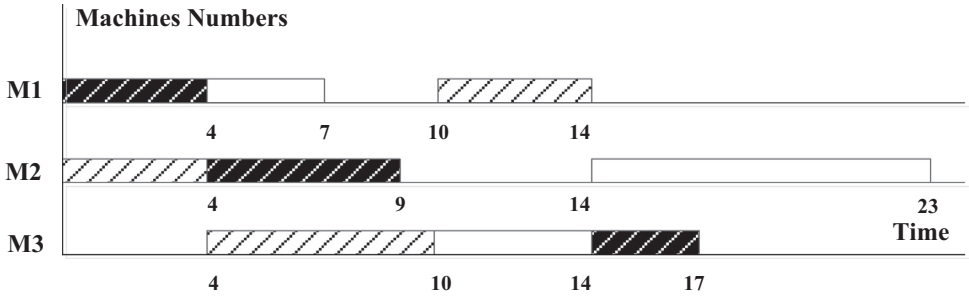

Fig. 1. A Gantt chart representation of a solution for the 3 × 3 problem.

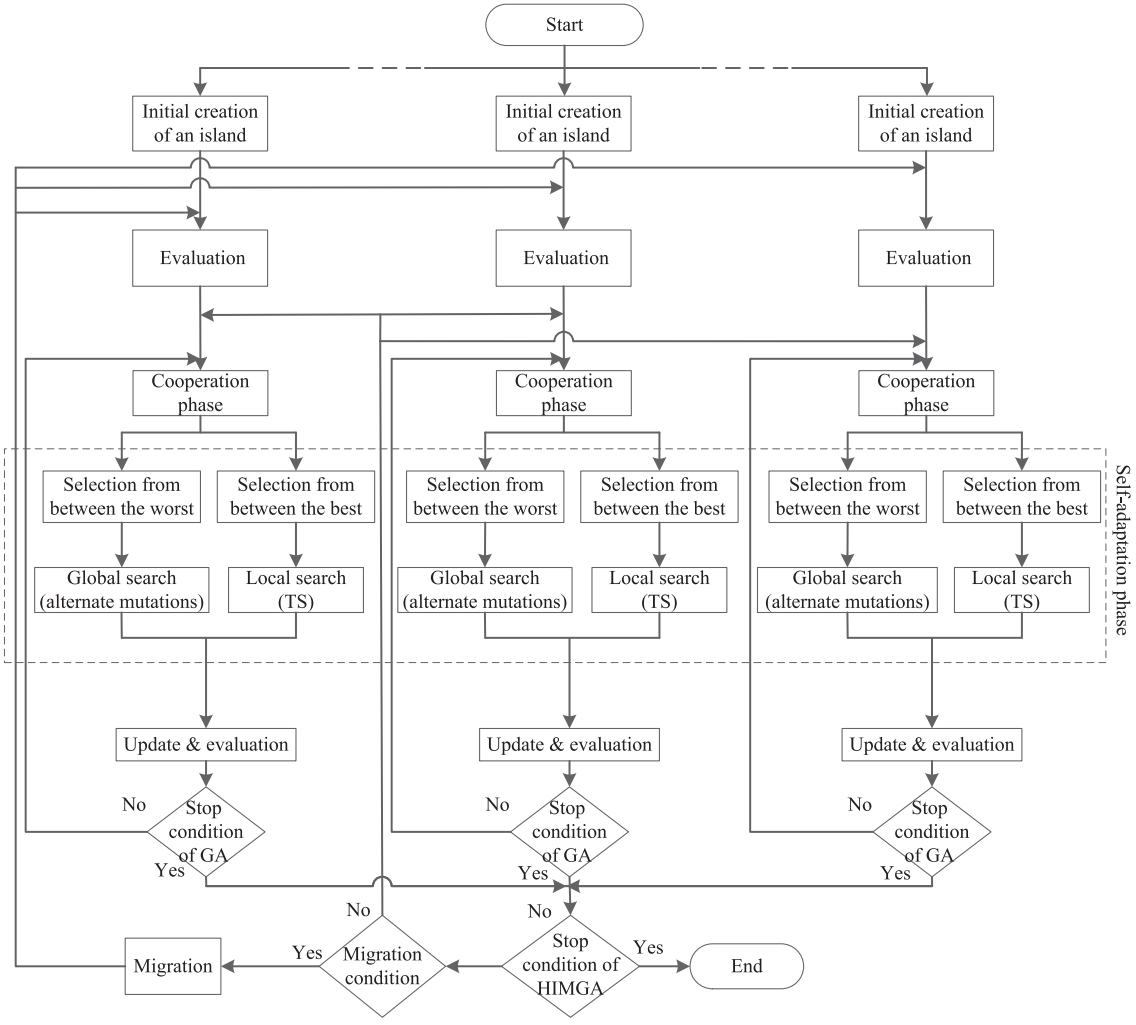

Fig. 2. The general framework of the HIMGA.

3.1.1. Communications topology structure

It specifies the migration routes between islands. Currently, the common topology structures are ring, star-shaped, mesh, unrestricted (complete net), and liner matrix. The communication cost depends mainly on the network diameter. Given that we intend to allow a direct and low cost migration from any island to another, we adopt the unrestricted structure (Man, Tang, & Kwong, 1999), shown in Fig. 3 in the case of 3 islands.

3.1.2. Migration rate

It specifies the number of migrants and the date of migration. In order to get a good convergence, migration should happen in a timely fashion: neither too early nor too late, usually when each island has converged (Engelbrecht, 2007).

In this paper, we let one migrant from each island to migrate to the others, if there are no improvements in all of the islands for five successive generations.在本文中，如果所有岛屿连续5代没有改善，我们让每个岛屿的一个移民迁移到其他岛屿

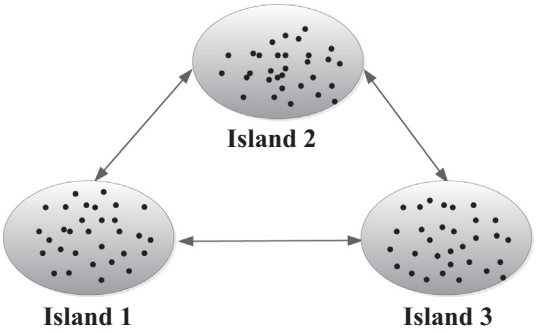

Fig. 3. The unrestricted topology structure.

3.1.3. Migration selection method

In order to allow islands to share their information about the promising regions in the search space, the roulette wheel method is adopted (Goldberg, 1989). In this method, the probability of an individual x_j to be selected from a population composed of n individuals for migration is proportional to its fitness $f(x_j)$, and can be calculated using this formula

$$P(x_j) = f(x_j) \Big/ \sum_{i=1}^{n} f(x_i).$$

3.1.4. Replacement method

The migration replacement method determines which natives will be removed from their island to make room for coming migrants. The replacement method used in this work specifies that the migrants replace the worst natives.

3.2. Fitness function

To determine the survival probability of an individual at the next generation, a fitness function is used to measure how good the solution represented by an individual is for the problem being considered. In this work, we use the well-known fitness function proposed by Goldberg (1989), which is defined by this formula $F(x) = C_{maxP} - C_{max}(x)$, where C_{maxP} is the biggest makespan value observed in the current population, and $C_{max}(x)$ is the makespan value of the individual x.

3.3. Chromosome representation and decoding

We use the preference list representation scheme proposed by Davis (1985), because it was shown by the study made by Ponnambalam, Aravindan, and Rao (2001) to be one of the best methods in terms of solution quality.

To represent a problem of the size $n \times m$, a chromosome is formed of m sub-chromosomes, each for one machine. Each sub-chromosome consists of a string of n symbols generated randomly, and each symbol represents an operation that should be processed on the related machine. For example, a chromosome for the 3×3 JSSP given in Table 1, may take the form [(123) (321) (213)]. Initially, sub-chromosomes do not describe the actual operations order on machines, because they are just preference lists and may produce infeasible individuals. The actual schedule is deduced from the chromosome through a simulation via the decoding procedure proposed by Della Croce, Tadei, and Volta (1995), and reformulated by Cheng et al. (1996), which analyzes the state of the waiting queues in front of the machines and if necessary uses the preference lists to determine the schedule, that is, the operation which appears first in the preference list will be chosen.

3.4. Initial islands

All islands have the same size; all individuals are generated randomly, because using other advanced strategies such as differential evolution (DE) algorithm (Zobolas, Tarantilis, & Ioannou, 2009), or G&T algorithm (Park et al., 2003) may provide better results, but will add undesirable complexity to the proposed hybridization model.

3.5. Cooperation phase and selection method

In this work, the selection method used for cooperation phase is also the roulette wheel (Goldberg, 1989).

Cooperation phase performs evolution through a cooperative behavior between individuals that exchange their information about the search space, using the crossover operator which produces new offspring that combine the characteristics of their parents.

Usual crossover operators are not shown to be effective on JSSP; for that reason, we use the order-based crossover (Qi et al., 2000). The mechanism of this crossover is described by first choosing two cut points at two random locations on the gene strings, then passing the parts between the two cut points to the two offspring. The operator then begins to construct the remaining left hand and right hand sides of each offspring, by going over its second parent and eliminating the same numbers with the passed part of its first parent, and filling up the missing left and right sides according to an order existing on its second parent.

The previous process will be done pair-wise. Fig. 4 illustrates this genetic operator.

3.6. The new self-adaptation phase strategy

Self-adaptation phase is usually used for providing search diversification and avoiding premature convergence, by allowing individuals to evolve (adapt to their environments) independently, using their own knowledge about the search space (their environment). This is usually accomplished by using a random mutation operator that probably makes the adapted individuals move for long distances (outside the current neighborhood) to new regions in the search space.

In this work, we propose a new naturally inspired self-adaptation phase strategy that is capable of providing search diversification and intensifications together. This is done by recruiting the best individuals for performing a local search using TS, and the worst ones for performing a global search using a combination of 3 classical random mutation operators that alternate randomly with equal probability. The proposed self-adaptation strategy imitates the nature in a better way, because in the nature, best individuals who are most

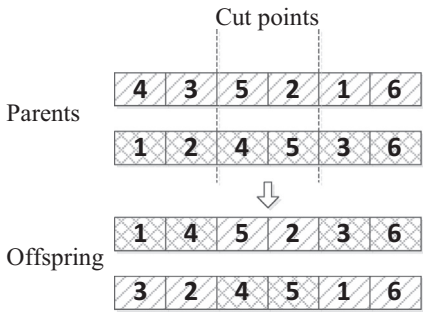

Fig. 4. Order-based crossover.

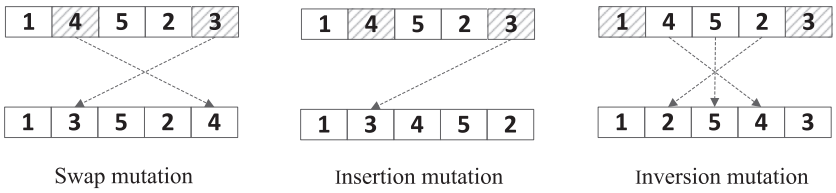

Fig. 5. Swap, insertion, and inversion mutation operators.

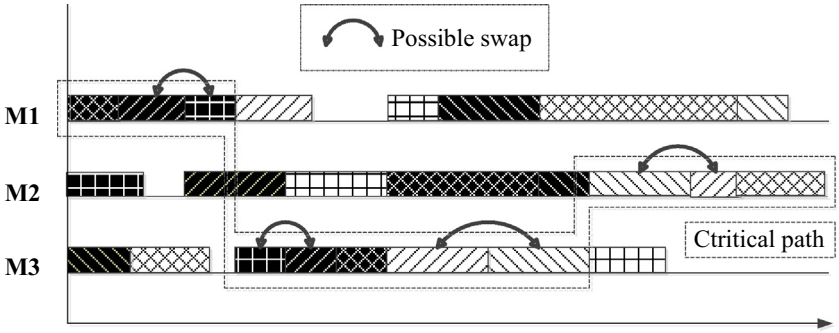

Fig. 6. Nowicki and Smutnicki's N5 neighborhood structure.

adapted to their environment will likely tend to search for a better location within short distances inside their neighborhoods, whereas worst individuals who are least adapted to their environments tend to search in new regions (neighborhoods), hoping in finding a better region to live in.

The 3 classical mutation operators used are inversion, insertion, and swap (Gen & Cheng, 1997). Inversion mutation selects two genes within a chromosome at random and then inverts the sub-string between these two genes. Insertion mutation selects a gene at random and then inserts it in a random location. Swap mutation selects two genes at random and then swaps the values assigned to them. An example of each is shown in Fig. 5.

3.7. Infeasible individuals

It is possible to get infeasible individuals during the initial generation of islands, or as a result of applying genetic operators. These individuals are usually handled by rejecting them, or accepting them with a penalty imposed on the fitness function, or developing a repairing procedure that converts them to feasible ones. The first and second methods are shown to be practically inefficient for JSSP, because the space of feasible individuals is very small compared to the space of possible individuals (Moraglio, Ten Eikelder, & Tadei, 2005); for that reason, we use the third method, by means of the decoding procedure introduced in Section 3.3, by starting from the earliest possible point of infeasibility (earliest mutated or cros gene), i.e. we keep the parts that precede that point as are (because they were kept without any changes) and reschedule the parts that succeed it.

3.8. Replacement strategies and stop condition

A steady state generation replacement with elitist strategy (Goldberg, 1989) is used. A new population does not totally replace the old one; offspring replace their parents; the best solution is kept without a change from one generation to next one.

The stop condition of HIMGA is either the best known solution has been found, or the maximum number of generations has been reached.

3.9. Tabu search and the neighborhood structure for JSSP

Tabu Search (TS), proposed by Glover (1989, 1990), is one of the most widespread LSMs for solving combinatorial optimization problems, and can be described as follows. TS explores the search space by performing a series of moves, in each move, the current solution is replaced by its best neighboring solution even if it is worse, which facilitates escaping from a local optimum. To avoid cycles, the moves that have been recently performed are banned by storing them on a short-term memory called tabu list. In addition, TS can maintain a long-term memory that stores various information about the history of the whole search process such as a fixed number of elite solutions, then restart strategies can be used mainly to strengthen its diversification capabilities, i.e. to identify new different promising regions in the search space; however, these features are not considered in the TS implementation adopted in this study, because they are replaced with IMGA.

A neighborhood structure is a mechanism that can obtain a new set of neighboring solutions by applying small modifications to a given solution, it plays a substantial role in the effectiveness and efficiency of TS, because TS iteratively proceeds from one neighbor to another in the search space; for that reason, a good one should be able to eliminate unnecessary and infeasible moves if that is possible, and also be restricted as much as possible, without exposing to the risk of narrow exploration which may prevent reaching global optimums. Many structures have been developed during the last 3 decades to fit the JSPP, most of them rely on the mechanism that changes the location of one or two operations that belong to the same critical block. In this research, we use the N5 structure proposed by Nowicki and Smutnicki (1996), which is shown in Fig. 6, and can be defined as follows. Consider a given critical path and divide it into several blocks, each of which contains at least two operations processed on the same machine. The neighborhood

Table 2
Results of HIMGAs with different self-adaptation strategies.

Problem			HIMGA			HIMGA-Tabu			HIMGA-Random			HIMGA-Mutation		
Name	Size	BKS	BS	AvS	AvT (s)	BS	AvS	AvT (s)	BS	AvS	AvT (s)	BS	AvS	AvT (s)
ft06	6 × 6	55	55	55.00	1.12	55	55.00	1.16	55	55.00	1.13	55	55.00	0.08
ft10	10 × 10	930	930	936.25	20.34	930	936.68	42.20	930	935.84	63.32	953	986.32	27.72
ft20	20 × 5	1165	1165	1169.16	15.96	1165	1168.12	28.40	1165	1168.80	29.92	1193	1234.52	30.28
orb01	10 × 10	1059	1059	1068.52	43.48	1059	1068.40	58.48	1059	1067.32	67.88	1086	1123.32	32.88
orb02	10 × 10	888	888	889.36	161.00	888	890.00	206.24	888	890.52	301.60	900	926.32	32.72
orb03	10 × 10	1005	1005	1016.92	135.68	1005	1016.12	197.33	1005	1015.52	141.36	1034	1078.00	32.96
orb04	10 × 10	1005	1005	1011.16	398.78	1005	1011.32	467.80	1005	1012.76	530.84	1032	1057.88	30.52
orb05	10 × 10	887	887	891.04	512.52	887	889.96	741.88	887	891.48	600.23	891	916.80	30.56
orb06	10 × 10	1010	1010	1016.48	203.92	1010	1015.72	316.68	1010	1016.16	312.21	1031	1051.84	30.44
orb07	10 × 10	397	397	399.72	12.12	397	399.48	16.00	397	399.56	13.12	416	444.60	33.85
orb08	10 × 10	899	899	909.20	38.88	899	910.04	86.21	899	909.08	27.68	932	955.88	31.28
orb09	10 × 10	934	934	934.00	18.28	934	938.08	21.54	934	939.68	31.32	948	971.80	30.92
orb10	10 × 10	944	944	946.80	15.36	944	948.76	34.11	944	946.20	17.04	959	989.48	30.52
ABZ5	10 × 10	1234	1234	1236.12	72.32	1234	1237.80	104.56	1234	1236.64	97.43	1238	1251.60	24.16
ABZ6	10 × 10	943	943	943.08	8.00	943	943.40	8.40	943	943.24	9.51	947	956.08	24.16
ABZ7	20 × 15	656	662	666.70	1641.67	666	668.89	2314.67	663	667.44	1785.32	706	720.96	54.72
ABZ8	20 × 15	665	676	678.00	1581.67	678	679.33	1831.67	678	679.38	2067.94	727	741.44	54.64
ABZ9	20 × 15	678	688	690.10	1438.19	690	691.75	1705.19	691	691.54	1897.05	753	772.36	55.76
la01	10 × 5	666	666	666.00	2.60	666	666.00	2.84	666	666.00	2.92	666	667.12	0.12
la02	10 × 5	655	655	655.00	3.16	655	655.04	3.83	655	655.00	4.11	655	668.60	7.24
la03	10 × 5	597	597	597.24	3.44	597	597.24	4.08	597	597.48	4.67	597	608.48	16.80
la04	10 × 5	590	590	590.12	3.04	590	590.00	3.12	590	600.48	5.56	590	600.48	7.32
la05	10 × 5	593	593	593.00	3.08	593	593.00	4.69	593	593.00	4.81	593	593.00	0.12
la06	15 × 5	926	926	926.00	6.20	926	926.00	7.24	926	926.00	8.12	926	926.00	0.16
la07	15 × 5	890	890	890.00	6.60	890	890.00	7.60	890	890.00	5.60	890	897.12	0.20
la08	15 × 5	863	863	863.00	5.00	863	863.00	6.60	863	863.00	5.32	863	863.00	30.04
la09	15 × 5	951	951	951.00	5.40	951	951.00	6.56	951	951.00	4.56	951	951.00	0.12
la10	15 × 5	958	958	958.00	6.11	958	958.00	6.62	958	958.00	6.80	958	958.00	0.12
la11	20 × 5	1222	1222	1222.00	14.44	1222	1222.00	14.72	1222	1222.00	15.73	1222	1222.00	0.08
la12	20 × 5	1039	1039	1039.00	11.08	1039	1039.00	12.84	1039	1039.00	13.72	1039	1039.00	0.20
la13	20 × 5	1150	1150	1150.00	13.16	1150	1150.00	13.40	1150	1150.00	15.04	1150	1150.00	0.12
la14	20 × 5	1292	1292	1292.00	13.20	1292	1292.00	12.20	1292	1292.00	14.20	1292	1292.00	0.20
la15	20 × 5	1207	1207	1207.00	15.04	1207	1207.00	16.97	1207	1207.00	16.68	1207	1224.80	4.08
la16	10 × 10	945	945	947.56	20.08	945	949.52	22.95	945	948.48	28.12	946	974.24	29.04
la17	10 × 10	784	784	784.12	18.36	784	784.04	17.47	784	784.00	17.80	784	792.16	19.00
la18	10 × 10	848	848	849.80	23.40	848	849.92	28.40	848	849.32	30.64	848	862.60	22.20
la19	10 × 10	842	842	844.16	28.64	842	844.24	39.64	842	843.36	46.56	842	862.80	24.64
la20	10 × 10	902	902	903.08	27.56	902	903.16	38.28	902	903.64	43.68	910	916.24	29.12
la21	15 × 10	1046	1046	1050.52	480.00	1046	1052.56	512.54	1046	1052.44	696.68	1097	1123.28	38.28
la22	15 × 10	927	927	931.92	364.13	927	932.12	671.28	927	932.72	697.96	964	988.20	38.72
la23	15 × 10	1032	1032	1032.00	16.00	1032	1032.00	16.40	1032	1032.00	15.32	1032	1044.84	9.88
la24	15 × 10	935	935	939.92	66.45	935	940.56	97.56	935	940.36	93.36	977	999.36	38.52
la25	15x10	977	977	983.72	177.31	977	981.48	252.12	977	982.58	217.68	1016	1043.16	38.96
la26	20 × 10	1218	1218	1218.00	35.40	1218	1218.00	37.60	1218	1218.04	44.12	1246	1280.12	45.48
la27	20 × 10	1235	1235	1247.80	380.54	1235	1245.96	578.20	1235	1246.56	450.00	1307	1334.96	42.12
la28	20 × 10	1216	1216	1216.00	66.00	1216	1218.84	66.48	1216	1220.12	97.04	1278	1308.56	42.28
la29	20 × 10	1152	1153	1163.04	1268.40	1162	1165.50	1527.86	1162	1163.80	1349.51	1263	1292.56	42.60
la30	20 × 10	1355	1355	1355.00	42.76	1355	1355.00	46.92	1355	1355.00	46.86	1355	1403.88	39.60
la31	30 × 10	1784	1784	1784.00	1026.00	1784	1784.00	1121.28	1784	1784.00	1108.52	1784	1824.12	18.20
la32	30 × 10	1850	1850	1850.00	112.40	1850	1850.00	123.80	1850	1850.00	126.40	1888	1908.56	7.96
la33	30 × 10	1719	1719	1719.00	106.32	1719	1719.00	121.32	1719	1719.00	176.84	1719	1766.08	12.68
la34	30 × 10	1721	1721	1721.00	122.72	1721	1721.00	140.00	1721	1721.00	171.35	1728	1750.80	62.28
la35	30 × 10	1888	1888	1888.00	99.20	1888	1888.00	132.20	1888	1888.00	169.80	1888	1899.28	59.76
la36	15 × 15	1268	1268	1276.68	876.16	1268	1279.40	1040.44	1268	1279.44	1027.15	1315	1347.72	44.00
la37	15 × 15	1397	1397	1407.92	1867.65	1397	1405.68	1983.08	1397	1412.68	2164.76	1456	1501.36	44.04
la38	15 × 15	1196	1196	1205.28	651.90	1196	1207.52	832.49	1196	1206.20	745.52	1254	1298.16	44.32
la39	15 × 15	1233	1233	1240.76	880.32	1233	1242.36	1931.68	1233	1241.76	2257.69	1268	1318.40	43.52
la40	15 × 15	1222	1224	1226.40	946.10	1224	1226.52	1011.66	1224	1226.72	1021.19	1257	1291.72	44.12
TA01	15 × 15	1231	1231	1231.00	934.44	1231	1245.28	1647.48	1231	1246.00	1830.16	1310	1329.20	38.68
TA11	20 × 15	1357	1366	1375.24	2473.00	1369	1377.91	2819.19	1370	1380.82	2919.05	1488	1535.64	48.72
TA21	20 × 20	1642	1650	1668.16	3437.00	1661	1669.43	3510.73	1662	1672.57	4034.50	1801	1841.40	62.20
YN1	20 × 20	884	893	904.08	2501.70	897	904.23	2683.41	899	902.85	2840.23	941	968.84	57.56
YN2	20 × 20	904	913	923.72	2731.70	915	924.60	2819.36	915	925.70	3976.23	957	998.00	56.52
YN3	20 × 20	892	900	906.96	3368.00	905	908.00	3751.01	904	907.60	3681.95	965	983.56	57.40
YN4	20 × 20	968	977	984.56	4205.00	974	985.38	4703.94	977	983.00	5391.48	1057	1076.48	58.32
SWV11	50 × 10	2983	3095	3163.00	17968.00	3133	3196.25	19167.29	3146	3214.87	20790.45	3587	3715.80	70.36

consists of all schedules that can be obtained from the current one by swapping the last two operations in the first block, the first two operations in the last block, and the first and last two operations in the internal blocks.

4. Computational results

To illustrate the effectiveness of the proposed algorithm HIMGA, we consider 76 instances from five classes of standard

Table 3
Average relative deviations of HIMGAs with different self-adaptation strategies.

Algorithm	NIS	BS-ARD		Improvement	AvS-ARD		Improvement
		OA (%)	NIMGA (%)	NIMGA (%)	OA (%)	NIMGA (%)	NIMGA (%)
HIMGA-Tabu	66	0.27	0.20	0.07	0.65	0.57	0.08
HIMGA-Random	66	0.28	0.20	0.08	0.69	0.57	0.12
HIMGA-Mutation	66	2.95	0.20	2.75	5.09	0.57	4.52

JSSP benchmark instances: Fisher and Thompson (1963) instances ft06, ft10, ft20; Lawrence (1984) instances la01–la40; and Applegate and Cook (1991) instances orb01–orb10; five of Adams et al. (1988) instances denoted as ABZ5–ABZ9; Yamada and Nakano (1992) instances YN1–YN4; three of Taillard (1993) instances denoted as TA01, TA11, TA21; eleven of Storer, Wu, and Vaccari (1992) instances denoted as SWV01–SWV11.

The algorithm was implemented in C++, and the tests were run on a PC with 3.40 GHz Intel(R) Core (TM) i7-3770 CPU and 8.00 GB. The parameters were tuned through a number of experiments; as a result, they were fixed as follows: the number of islands 3, the island size 100, the maximum number of generations for the HIMGA 100, the maximum number of generations for the classical GA 1000, crossover probability 0.8, mutation probability 0.05, migration rate 1 individual after each five successive generations without any improvement in any island, tabu list length 12, maximum number of moves without improvement 2000.

4.1. Experiment 1 – evaluation of the proposed self-adaptation phase strategy

In this experiment, HIMGA was compared with other three versions of it that are identical, except that they use alternative self-adaptation phase strategies: an HIMGA that recruits randomly selected individuals to do local search using TS and global search using mutation operators (HIMGA-Random); an HIMGA that only recruits the best individuals to do local search using TS (HIMGA-Tabu); and an HIMGA that only recruits the worst individuals to do global search using mutation operators (HIMGA-Mutation).

Table 2 shows the experimental results on 66 instances, it lists problem name, problem size (number of jobs × number of machines), the best known solution (BKS); the best solution (BS), the average solution (AvS), and the average computation time (in seconds) AvT(s) obtained from 10 independent runs by each of the compared algorithms. From Table 2, it can be seen that HIMGA is capable of obtaining better best and average solutions in less computational time than all the others on almost all the instances; however, to make a wider comparison, we calculated the relative deviation of the best solution by this formula BS-RD = 100 × (BS-BKS)/BKS, and the relative deviation of the average solution by this formula AvS-RD = 100 × (AvS-BKS)/BKS. The average values of BS-RD and AvS-RD denoted as BS-ARD and AvS-ARD were also calculated for each algorithm over all instances.

Table 3 shows the number of instances solved (NIS), BS-ARD and AvS-ARD values for HIMGA and the other algorithms (OA), and the improvement achieved by HIMGA in BS-ARD and AvS-ARD values with respect to each of the other algorithms. From Table 3, it can be seen that HIMGA yields improvements with respect to all compared algorithms, which validates the effectiveness of the proposed self-adaptation phase strategy.

4.2. Experiments 2 – comparing HIMGA with other works

In this experiment, HIMGA was also run 10 times for each of the instances SWV01–SWV10; the numerical results on the 76 instances were compared with other 15 recent works found in the literature:

- Tabu with path relinking TS/PR (Peng et al., 2015).
- Hybrid evolutionary algorithm (EA) with LSMs
  - EA with TS **HEA** (Cheng et al., 2013).
  - DE with TS **DE-TS** (Ponsich & Coello, 2013)
  - GA with TS **GATS** (Meeran & Morshed, 2014).
  - Asexual GA with TS and GT algorithms **TGA** (Amirghasemi & Zamani, 2015).
  - GA with a LSM based on the N5 neighborhood **HGA** (Gonçalves, de Magalhães Mendes, & Resende, 2005).
  - GA with a new LSM **HGA** (Qing-dao-er-ji & Wang, 2012).
  - Agent Based GA with a new LSM **aLSGA** (Asadzadeh, 2015).
  - Memetic algorithm **MA** (Gao et al., 2011).
  - Biogeography-based optimization with chaos theory **HBBO** (Wang & Duan, 2014).
  - GA with multi-parents crossover hybridized with a new LSM **EPPX** (Moin, Chung Sin, & Omar, 2014).
  - GA with DE and VNS **DRGA$_{VNS}$** (Zobolas et al., 2009).
- Hybrid IMGA methods
  - Hybrid IMGA based on asynchronous colony GA (ACGA) and autonomous immigration GA (AIGA) **HPGA** (Yusof et al., 2011).
  - IMGA with VNS **PLSGA** (Asadzadeh, 2014).
  - IMGA with pair-wise interchange method **MP-HGA** (Kalantari & SanieeAbadeh, 2013).

Table 4 shows the comparison with the other works whose average solutions were not reported on 66 instances. From Table 4, it can be noticed that HIMGA obtains the best known solutions for 54 instances, i.e. in about 82% of the instances, and its results are either equal or better than all the other algorithms on the whole set of instances, except 3 instances: ABZ7 and ABZ8 when they were solved by DRGA$_{VNS}$, and ABZ9 when it was solved by GATS.

Table 5 does the same, but when the average solutions were reported on 72 instances. From Table 5, it can be noted that HIMGA obtains the best known solutions for 53 instances, i.e. in about 74% of the instances, and its performance is less competitive than before, because it sometimes obtains worse results with respect to some of the other works. To make a precise comparison, BS-ARD and AvS-ARD were calculated for HIMGA and the 15 compared algorithms. As a result, HIMGA could score a BS-ARD of 0.3% and an AvS-ARD of 0.75% on the 76 instances; however, to make a fair comparison, we re-calculated ARDs for it on each sub-set of the instances considered by the compared algorithms.

Table 6 shows the number of instances solved (NIS), the BS-ARD and AvS-ARD values for HIMGA and the other algorithms (OA), and the improvement achieved by HIMGA with respect to each of the other algorithms. From Table 6, it can be seen that HIMGA yields

Table 4
Comparison of the best solutions of HIMGA with other works.

Problem		HIMGA		Gonçalves et al. (2005) HGA	Zobolas et al. (2009) $DRGA_{VNS}$	Gao et al. (2011) MA	Yusof et al. (2011) HPGA	Qing-dao-er-ji and Wang (2012) HGA	Kalantari and SanieeAbadeh (2013) MP-HGA	Wang and Duan (2014) HBBO	Meeran and Morshed (2014) GATS	Moin et al. (2014) EPPX	Asadzadeh (2014) PLSGA
Name	Size	BKS	BS	BS	BS	BS	BS	BS	BS	BS	BS	BS	BS
ft06	6 × 6	55	55	55	55	55	–	55	–	55	55	55	–
ft10	10 × 10	930	930	930	930	930	931	930	–	930	930	930	–
ft20	20 × 5	1165	1165	1165	1165	1165	1165	1165	–	1165	1165	1178	–
orb01	10 × 10	1059	1059	–	1059	–	1085	–	–	1059	1059	1077	–
orb02	10 × 10	888	888	–	888	–	–	–	–	919	888	889	–
orb03	10 × 10	1005	1005	–	1005	–	–	–	–	–	1005	1022	–
orb04	10 × 10	1005	1005	–	1005	–	1054	–	–	–	1005	1005	–
orb05	10 × 10	887	887	–	887	–	–	–	–	–	887	890	–
orb06	10 × 10	1010	1010	–	1010	–	–	–	–	–	1010	1021	–
orb07	10 × 10	397	397	–	397	–	–	–	–	–	397	397	–
orb08	10 × 10	899	899	–	899	–	–	–	–	–	899	899	–
orb09	10 × 10	934	934	–	934	–	–	–	–	–	934	934	–
orb10	10 × 10	944	944	–	944	–	–	–	–	–	944	944	–
ABZ5	10 × 10	1234	1234	–	1234	–	–	–	–	1250	1234	1234	–
ABZ6	10 × 10	943	943	–	943	–	–	–	–	948	943	943	–
ABZ7	20 × 15	656	662	–	659	–	–	–	–	–	668	683	–
ABZ8	20 × 15	665	676	–	670	–	–	–	–	–	682	701	–
ABZ9	20 × 15	678	688	–	688	–	–	–	–	734	679	712	–
la01	10 × 5	666	666	666	–	666	–	666	666	666	666	–	666
la02	10 × 5	655	655	655	655	655	680	655	655	655	655	–	655
la03	10 × 5	597	597	597	–	597	–	597	621	597	597	–	597
la04	10 × 5	590	590	590	–	590	–	590	602	590	590	–	–
la05	10 × 5	593	593	593	–	593	–	593	593	593	593	–	–
la06	15 × 5	926	926	926	–	926	–	926	926	926	926	–	–
la07	15 × 5	890	890	890	–	890	–	890	890	890	890	–	–
la08	15 × 5	863	863	863	–	863	–	863	863	863	863	–	863
la09	15 × 5	951	951	951	–	951	–	951	951	951	951	–	951
la10	15 × 5	958	958	958	–	958	–	958	958	958	958	–	958
la11	20 × 5	1222	1222	1222	–	1222	–	1222	1222	1222	1222	–	–
la12	20 × 5	1039	1039	1039	–	1039	–	1039	1039	1039	1039	–	1039
la13	20 × 5	1150	1150	1150	–	1150	–	1150	1150	1150	1150	–	–
la14	20 × 5	1292	1292	1292	–	1292	–	1292	1292	1292	1292	–	1292
la15	20 × 5	1207	1207	1207	–	1207	–	1207	1207	1207	1207	–	1207
la16	10 × 10	945	945	945	945	945	947	945	–	945	945	–	945
la17	10 × 10	784	784	784	–	784	–	784	–	784	784	–	784
la18	10 × 10	848	848	848	–	848	–	848	–	848	848	–	848
la19	10 × 10	842	842	842	842	842	–	844	–	842	842	–	842
la20	10 × 10	902	902	907	–	902	–	907	–	902	902	–	907
la21	15 × 10	1046	1046	1046	1046	1055	1067	1046	–	1046	1047	–	1046
la22	15 × 10	927	927	935	927	927	–	935	–	933	927	–	960
la23	15 × 10	1032	1032	1032	–	1032	–	1032	–	1032	1032	–	1032
la24	15 × 10	935	935	953	935	940	–	953	–	935	935	–	–
la25	15 × 10	977	977	986	977	984	–	981	–	977	977	–	–
la26	20 × 10	1218	1218	1218	–	1218	–	1218	–	1218	–	–	1218
la27	20 × 10	1235	1235	1256	1236	1261	–	1236	–	–	–	–	1281
la28	20 × 10	1216	1216	1232	1224	1216	–	1216	–	–	–	–	–
la29	20 × 10	1152	1153	1196	1160	1190	1154	1160	–	–	–	–	–
la30	20 × 10	1355	1355	1355	1355	1355	–	1355	–	–	–	–	1355
la31	30 × 10	1784	1784	1784	–	1784	1906	1784	–	1784	–	–	1784
la32	30 × 10	1850	1850	1850	–	1850	V	1850	–	–	–	–	1850
la33	30 × 10	1719	1719	1719	–	1719	–	1719	–	–	–	–	1719
la34	30 × 10	1721	1721	1721	–	1721	–	1721	–	–	–	–	–
la35	30 × 10	1888	1888	1888	–	1888	–	1888	–	–	–	–	–
la36	15 × 15	1268	1268	1279	1268	1281	1308	1287	–	1268	–	–	–
la37	15 × 15	1397	1397	1408	1408	1431	–	1407	–	–	–	–	–
la38	15 × 15	1196	1196	1219	1202	1216	–	1196	–	–	–	–	–
la39	15 × 15	1233	1233	1246	1233	1241	–	1233	–	–	–	–	–
la40	15 × 15	1222	1224	1241	1229	1233	–	1229	–	–	–	–	––
TA01	15 × 15	1231	1231	–	–	–	–	–	–	–	1314	–	–
TA11	20 × 15	1357	1366	–	–	–	–	–	–	–	1442	–	–
TA21	20 × 20	1642	1650	–	–	–	–	–	–	–	1675	–	–
YN1	20 × 20	884	893	–	909	–	–	–	–	–	–	911	–
YN2	20 × 20	904	913	–	–	–	–	–	–	–	–	941	–
YN3	20 × 20	892	900	–	–	–	–	–	–	–	–	928	–
YN4	20 × 20	968	977	–	–	–	–	–	–	–	–	1011	–
SWV11	50 × 10	2983	3095	–	3206	–	–	–	–	–	–	–	–

Table 5
Comparison of the best and average solutions of HIMGA with other works.

Problem			HIMGA		Cheng et al. (2013) HEA		Peng et al. (2015) TS/PR		Asadzadeh (2015) aLSGA	Amirghasemi and Zamani (2015) TGA	
Name	Size	BKS	BS	AvS	BS	AvS	BS	AvS	BS	BS	AvS
ft06	6 × 6	55	55	55.00	55	55.00	55	55.00	55	55	55.00
ft10	10 × 10	930	930	936.25	930	930.00	930	930.00	930	930	930.00
ft20	20 × 5	1165	1165	1169.16	1165	1165.00	1165	1165.00	1165	1165	1165.00
orb01	10 × 10	1059	1059	1068.52	–	–	1059	1059.00	1092	1059	1060.80
orb02	10 × 10	888	888	889.36	–	–	888	888.00	894	888	888.00
orb03	10 × 10	1005	1005	1016.92	–	–	1005	1005.00	1029	1005	1010.70
orb04	10 × 10	1005	1005	1011.16	–	–	1005	1005.00	1016	1005	1010.60
orb05	10 × 10	887	887	891.04	–	–	887	887.00	901	887	887.00
orb06	10 × 10	1010	1010	1016.48	–	–	1010	1010.00	1028	1010	1012.20
orb07	10 × 10	397	397	399.72	–	–	397	397.00	405	397	397.00
orb08	10 × 10	899	899	909.20	–	–	899	899.00	914	899	899.70
orb09	10 × 10	934	934	934.00	–	–	934	934.00	943	934	936.70
orb10	10 × 10	944	944	946.80	–	–	944	944.00	–	944	944.00
la01	10 × 5	666	666	666.00	666	666.00	666	666.00	666	666	666.00
la02	10 × 5	655	655	655.00	655	655.00	655	655.00	655	655	655.00
la03	10 × 5	597	597	597.24	597	597.00	597	597.00	606	597	597.00
la04	10 × 5	590	590	590.12	590	590.00	590	590.00	593	590	590.00
la05	10 × 5	593	593	593.00	593	593.00	593	593.00	593	593	593.00
la06	15 × 5	926	926	926.00	926	926.00	926	926.00	926	926	926.00
la07	15 × 5	890	890	890.00	890	890.00	890	890.00	890	890	890.00
la08	15 × 5	863	863	863.00	863	863.00	863	863.00	863	863	863.00
la09	15 × 5	951	951	951.00	951	951.00	951	951.00	951	951	951.00
la10	15 × 5	958	958	958.00	958	958.00	958	958.00	958	958	958.00
la11	20 × 5	1222	1222	1222.00	1222	1222.00	1222	1222.00	1222	1222	1222.00
la12	20 × 5	1039	1039	1039.00	1039	1039.00	1039	1039.00	1039	1039	1039.00
la13	20 × 5	1150	1150	1150.00	1150	1150.00	1150	1150.00	1150	1150	1150.00
la14	20 × 5	1292	1292	1292.00	1292	1292.00	1292	1292.00	1292	1292	1292.00
la15	20 × 5	1207	1207	1207.00	1207	1207.00	1207	1207.00	1207	1207	1207.00
la16	10 × 10	945	945	947.56	945	945.00	945	945.00	946	945	945.00
la17	10 × 10	784	784	784.12	784	784.00	784	784.00	784	784	784.00
la18	10 × 10	848	848	849.80	848	848.00	848	848.00	848	848	848.00
la19	10 × 10	842	842	844.16	842	842.00	842	842.00	852	842	842.00
la20	10 × 10	902	902	903.08	902	902.00	902	902.00	907	902	902.00
la21	15 × 10	1046	1046	1050.52	1046	1046.00	1046	1046.00	1068	1046	1049.70
la22	15 × 10	927	927	931.92	927	927.00	927	927.00	956	927	927.00
la23	15 × 10	1032	1032	1032.00	1032	1032.00	1032	1032.00	1032	1032	1032.00
la24	15 × 10	935	935	939.92	935	935.00	935	935.00	966	935	935.30
la25	15 × 10	977	977	983.72	977	977.00	977	977.00	1002	977	979.00
la26	20 × 10	1218	1218	1218.00	1218	1218.00	1218	1218.00	1223	1218	1218.00
la27	20 × 10	1235	1235	1247.80	1235	1235.00	1235	1235.00	1281	1235	1236.00
la28	20 × 10	1216	1216	1216.00	1216	1216.00	1216	1216.00	1245	1216	1216.00
la29	20 × 10	1152	1153	1163.04	1153	1153.00	1153	1153.00	1230	1153	1166.60
la30	20 × 10	1355	1355	1355.00	1355	1355.00	1355	1355.00	1355	1355	1355.00
la31	30 × 10	1784	1784	1784.00	1784	1784.00	1784	1784.00	1784	1784	1784.00
la32	30 × 10	1850	1850	1850.00	1850	1850.00	1850	1850.00	1850	1850	1850.00
la33	30 × 10	1719	1719	1719.00	1719	1719.00	1719	1719.00	1719	1719	1719.00
la34	30 × 10	1721	1721	1721.00	1721	1721.00	1721	1721.00	1721	1721	1721.00
la35	30 × 10	1888	1888	1888.00	1888	1888.00	1888	1888.00	1888	1888	1888.00
la36	15 × 15	1268	1268	1276.68	1268	1268.00	1268	1268.00	–	1268	1268.00
la37	15 × 15	1397	1397	1407.92	1397	1397.00	1397	1397.00	–	1397	1399.50
la38	15 × 15	1196	1196	1205.28	1196	1196.00	1196	1196.00	–	1196	1197.70
la39	15 × 15	1233	1233	1240.76	1233	1233.00	1233	1233.00	–	1233	1233.70
la40	15 × 15	1222	1224	1226.40	1222	1222.60	1222	1222.00	–	1224	1225.50
ABZ5	10 × 10	1234	1234	1236.12	–	–	–	–	–	1234	1234.60
ABZ6	10 × 10	943	943	943.08	–	–	–	–	–	943	943.00
ABZ7	20 × 15	656	662	666.70	657	657.60	657	657.10	–	659	664.90
ABZ8	20 × 15	665	676	678.00	667	667.90	667	667.80	–	667	674.60
ABZ9	20 × 15	678	688	690.10	678	678.00	678	678.00	–	678	686.70
SWV01	20 × 10	1407	1420	1427.40	1407	1411.20	1407	1411.40	–	1421	1463.40
SWV02	20 × 10	1475	1480	1486.90	1475	1475.00	1475	1475.10	–	1475	1505.20
SWV03	20 × 10	1398	1420	1427.50	1398	1398.40	1398	1398.90	–	1423	1437.10
SWV04	20 × 10	1470	1480	1489.50	1470	1471.60	1470	1473.50	–	1508	1543.40
SWV05	20 × 10	1424	1426	1439.60	1425	1427.10	1425	1426.00	–	1445	1486.00
SWV06	20 × 15	1671	1696	1710.80	1672	1678.90	1671	1675.90	–	1722	1753.70
SWV07	20 × 15	1594	1619	1633.70	1595	1598.90	1595	1605.00	–	1634	1668.40
SWV08	20 × 15	1752	1787	1800.80	1752	1764.60	1752	1760.40	–	1804	1829.00
SWV09	20 × 15	1655	1661	1689.90	1657	1660.90	1655	1661.80	–	1691	1722.30
SWV10	20 × 15	1743	1761	1794.50	1743	1754.70	1743	1756.60	–	1780	1818.30
YN1	20 × 20	884	893	904.08	884	886.10	884	885.50	–	886	893.90
YN2	20 × 20	904	913	923.72	904	906.50	904	907.70	–	911	918.70
YN3	20 × 20	892	900	906.96	892	894.30	892	893.80	–	897	902.80
YN4	20 × 20	968	977	984.56	968	969.25	968	969.10	–	975	986.90

Table 6
Average relative deviations of HIMGA and the 15 compared works.

Algorithm	NIS	BS-ARD		Improvement	AvS-ARD		Improvement
		OA (%)	HIMGA (%)	HIMGA (%)	OA (%)	HIMGA (%)	HIMGA (%)
Asadzadeh (2015) aLSGA	47	0.93	0.00	0.93	–	–	–
Gonçalves et al. (2005) HGA	43	0.40	0.01	0.39	–	–	–
Zobolas et al. (2009) $DRGA_{VNS}$	36	0.45	0.25	0.20	–	–	–
Gao et al. (2011) MA	43	0.33	0.01	0.32	–	–	–
Yusof et al. (2011) HIMGA	10	2.36	0.01	2.35	–	–	–
Qing-dao-er-ji and Wang (2012) HGA	43	0.18	0.01	0.17	–	–	–
Kalantari and SanieeAbadeh (2013) MP-HGA	15	0.40	0.00	0.40	–	–	–
Wang and Duan (2014) HBBO	36	0.40	0.04	0.36	–	–	–
Meeran and Morshed (2014) GATS	46	0.43	0.11	0.32	–	–	–
Moin et al. (2014) EPPX	22	1.65	0.36	1.29	–	–	–
Asadzadeh (2014) PLSGA	23	0.34	0.00	0.34	–	–	–
Amirghasemi and Zamani (2015) TGA	72	0.32	0.25	0.07	0.77	0.67	0.10
Ponsich and Coello (2013) DE-TS	54	2.21	0.08	2.13	3.05	0.41	2.64
Peng et al. (2015) TS/PR	70	0.01	0.26	−0.25	0.07	0.69	−0.62
Cheng et al. (2013) HEA	60	0.01	0.30	−0.29	0.08	0.70	−0.62

improvements with respect to all compared algorithms, except TS/PR (Peng et al., 2015) and HEA (Cheng et al., 2013), which validates the effectiveness of the proposed algorithm HIMGA.

5. Conclusion

In this paper, we have proposed a new hybrid island model genetic algorithm HIMGA for minimizing the makespan in JSSP. To improve the effectiveness of IMGA and make it able to strike a better balance between diversification and intensification of the search process, a new naturally inspired self-adaptation phase strategy is proposed, in which the best individuals are recruited to perform a local search using tabu search (TS), and the worst ones are recruited to perform a global search using a combination of 3 classical random mutation operators. The HIMGA has been tested on 76 benchmarks instance, with the proposed self-adaptation phase strategy, and without it using the classical alternatives, and also compared with other 15 algorithms reported recently in the literature. Computational results show that HIMGA was able to find the best known solutions for about 71% of the 76 tested problem instances, and produce best and average solutions with average relative deviations of 0.3% and 0.75%, respectively, from the best known solutions. Computational results also show that HIMGA outperforms all other HIMGA versions that use alternative self-adaptation strategies, and also outperforms almost all the compared algorithms, which demonstrates that HIMGA can be considered as an effective method for solving JSSP.

Acknowledgment

This work was supported by the Institute of International Education.

References

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science, 34*(3), 391–401. http://dx.doi.org/10.1287/mnsc.34.3.391.

Amirghasemi, M., & Zamani, R. (2015). An effective asexual genetic algorithm for solving the job shop scheduling problem. *Computers & Industrial Engineering, 83*, 123–138. http://dx.doi.org/10.1016/j.cie.2015.02.011.

Applegate, D., & Cook, W. (1991). A computational study of the job shop scheduling problem. *ORSA Jornal on Computing, 3*(2), 149–156. http://dx.doi.org/10.1287/ijoc.3.2.149.

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

Asadzadeh, L. (2014). Solving the job shop scheduling problem with a parallel and agent-based local search genetic algorithm. *Journal of Theoretical and Applied Information Technology, 62*(2).

Asadzadeh, L., & Zamanifar, K. (2010). An agent-based parallel approach for the job shop scheduling problem. *Mathematical and Computer Modelling, 52*(11–12), 1957–1965. http://dx.doi.org/10.1016/j.mcm.2010.04.019.

Blackstone, J. H., Phillips, D. T., & Hogg, G. L. (1982). A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research, 20*(1), 27–45. http://dx.doi.org/10.1080/00207548208947745.

Carlier, J., & Pinson, E. (1989). An algorithm for solving the job-shop problem. *Management Science, 35*(2), 164–176. http://dx.doi.org/10.1287/mnsc.35.2.164.

Cheng, R., Gen, M., & Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation. *Computers & Industrial Engineering, 30*(4), 983–997. http://dx.doi.org/10.1016/0360-8352(96)00047-2.

Cheng, R., Gen, M., & Tsujimura, Y. (1999). A tutorial survey of job-shop scheduling problems using genetic algorithms, Part II: Hybrid genetic search strategies. *Computers & Industrial Engineering, 36*(2), 343–364. http://dx.doi.org/10.1016/S0360-8352(99)00136-9.

Cheng, T. C. E., Peng, B., & Lü, Z. (2013). A hybrid evolutionary algorithm to solve the job shop scheduling problem. *Annals of Operations Research*, 1–15. http://dx.doi.org/10.1007/s10479-013-1332-5.

Davis, L. (1985). Job shop scheduling with genetic algorithms. In: *Proceedings of an international conference on genetic algorithms and their applications* (pp. 136–140). Carnegie-Mellon University, Pittsburgh, PA, USA.

Della Croce, F., Tadei, R., & Volta, G. (1995). A genetic algorithm for the job shop problem. *Computers & Operations Research, 22*(1), 15–24. http://dx.doi.org/10.1016/0305-0548(93)E0015-L.

Engelbrecht, A. P. (2007). *Computational intelligence: An introduction* (2nd ed.). West Sussex, England: John Wiley & Sons.

Fisher, H., & Thompson, G. L. (1963). Probabilistic learning combinations of local job-shop scheduling rules. In J. Muth & G. Thompson (Eds.), *Industrial scheduling* (pp. 225–251). Englewood Cliffs, NJ: Prentice-Hall.

Fığlalı, N., Özkale, C., Engin, O., & Fığlalı, A. (2009). Investigation of ant system parameter interactions by using design of experiments for job-shop scheduling problems. *Computers & Industrial Engineering, 56*(2), 538–559. http://dx.doi.org/10.1016/j.cie.2007.06.001.

Gao, L., Zhang, G., Zhang, L., & Li, X. (2011). An efficient memetic algorithm for solving the job shop scheduling problem. *Computers & Industrial Engineering, 60*(4), 699–705. http://dx.doi.org/10.1016/j.cie.2011.01.003.

Gen, M., & Cheng, R. (1997). *Genetic algorithms and engineering design*. New York: John Wiley & Sons.

Glover, F. (1989). Tabu search – Part I. *ORSA Journal on Computing, 1*(3), 190–206. http://dx.doi.org/10.1287/ijoc.1.3.190.

Glover, F. (1990). Tabu search – Part II. *ORSA Journal on Computing, 2*(1), 4–32. http://dx.doi.org/10.1287/ijoc.2.1.4.

Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning*. MA: Addison-Wesley.

Graham, R. L., Lawler, E., Lenstra, J., & Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics, 5*, 287–326. http://dx.doi.org/10.1016/S0167-5060(08)70356-X.

Gonçalves, J. F., de Magalhães Mendes, J. J., & Resende, M. G. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research, 167*(1), 77–95. http://dx.doi.org/10.1016/j.ejor.2004.03.012.

Gu, J., Gu, X., & Gu, M. (2009). A novel parallel quantum genetic algorithm for stochastic job shop scheduling. *Journal of Mathematical Analysis and Applications, 355*(1), 63–81. http://dx.doi.org/10.1016/j.jmaa.2008.12.065.

Hertz, A., & Kobler, D. (2000). A framework for the description of evolutionary algorithms. *European Journal of Operational Research, 126*(1), 1–12. http://dx.doi.org/10.1016/S0377-2217(99)00435-X.

Jain, A. S., & Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research, 113*(2), 390–434. http://dx.doi.org/10.1016/S0377-2217(98)00113-1.

Kalantari, S., & SanieeAbadeh, M. (2013). An effective multi-population based hybrid genetic algorithm for job shop scheduling problem. *Bulletin of Electrical Engineering and Informatics, 2*(1), 59–64.

Konfrst, Z. (2004). Parallel genetic algorithms: Advances, computing trends, applications and perspectives. In *Parallel and distributed processing symposium, 2004. Proceedings. 18th International*. http://dx.doi.org/10.1109/IPDPS.2004.1303155.

Lageweg, B. J., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1977). Job-shop scheduling by implicit enumeration. *Management Science, 24*(4), 441–450. http://dx.doi.org/10.1287/mnsc.24.4.441.

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

Lian, Z., Jiao, B., & Gu, X. (2006). A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan. *Applied Mathematics and Computation, 183*(2), 1008–1017. http://dx.doi.org/10.1016/j.amc.2006.05.168.

Lin, T. L., Horng, S. J., Kao, T. W., Chen, Y. H., Run, R. S., Chen, R. J., et al. (2010). An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications, 37*(3), 2629–2636. http://dx.doi.org/10.1016/j.eswa.2009.08.015.

Man, K. F., Tang, K. S., & Kwong, S. (1999). *Genetic algorithms: Concepts and designs*. London: Springer. http://dx.doi.org/10.1007/978-1-4471-0577-0.

Meeran, S., & Morshed, M. S. (2014). Evaluation of a hybrid genetic tabu search framework on job shop scheduling benchmark problems. *International Journal of Production Research, 52*(19), 1–19. http://dx.doi.org/10.1080/00207543.2014.911417.

Moin, N. H., Chung Sin, O., & Omar, M. (2014). Hybrid genetic algorithm with multiparents crossover for job shop scheduling problems. *Mathematical Problems in Engineering*. Article ID 210680 (in press). http://dx.doi.org/10.1155/2015/210680.

Moraglio, A., Ten Eikelder, H., & Tadei, R. (2005). Genetic local search for job shop scheduling problem. Technical report CSM-435, University of Essex, UK.

Nowicki, E., & Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Management Science, 42*(6), 797–813. http://dx.doi.org/10.1287/mnsc.42.6.797.

Ombuki, B. M., & Ventresca, M. (2004). Local search genetic algorithms for the job shop scheduling problem. *Applied Intelligence, 21*(1), 99–109. http://dx.doi.org/10.1023/B:APIN.0000027769.48098.91.

Park, B. J., Choi, H. R., & Kim, H. S. (2003). A hybrid genetic algorithm for the job shop scheduling problems. *Computers & Industrial Engineering, 45*(4), 597–613. http://dx.doi.org/10.1016/S0360-8352(03)00077-9.

Peng, B., Lü, Z., & Cheng, T. C. E. (2015). A tabu search/path relinking algorithm to solve the job shop scheduling problem. *Computers & Operations Research, 53*, 154–164. http://dx.doi.org/10.1016/j.cor.2014.08.006.

Ponnambalam, S. G., Aravindan, P., & Rao, P. S. (2001). Comparative evaluation of genetic algorithms for job-shop scheduling. *Production Planning & Control: The Management of Operations, 12*(6), 560–574. http://dx.doi.org/10.1080/09537280175039768O.

Ponsich, A., & Coello, C. A. C. (2013). A hybrid differential evolution—Tabu search algorithm for the solution of job-shop scheduling problems. *Applied Soft Computing, 13*(1), 462–474. http://dx.doi.org/10.1016/j.asoc.2012.07.034.

Qi, J. G., Burns, G. R., & Harrison, D. K. (2000). The application of parallel multipopulation genetic algorithms to dynamic job-shop scheduling. *The International Journal of Advanced Manufacturing Technology, 16*(8), 609–615. http://dx.doi.org/10.1007/s001700070052.

Qing-dao-er-ji, R., & Wang, Y. (2012). A new hybrid genetic algorithm for job shop scheduling problem. *Computers & Operations Research, 39*(10), 2291–2299. http://dx.doi.org/10.1016/j.cor.2011.12.005.

Satake, T., Morikawa, K., Takahashi, K., & Nakamura, N. (1999). Simulated annealing approach for minimizing the makespan of the general job-shop. *International Journal of Production Economics, 60–61*, 515–522. http://dx.doi.org/10.1016/S0925-5273(98)00171-6.

Storer, R. H., Wu, S. D., & Vaccari, R. (1992). New search spaces for sequencing problems with application to job shop scheduling. *Management Science, 38*(10), 1495–1509. http://dx.doi.org/10.1287/mnsc.38.10.1495.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64*(2), 278–285. http://dx.doi.org/10.1016/0377-2217(93)90182-M.

Taillard, E. D. (1994). Parallel taboo search techniques for the job shop scheduling problem. *ORSA Journal on Computing, 6*(2), 108–117. http://dx.doi.org/10.1287/ijoc.6.2.108.

Tamilarasi, A., & kumar, T. A. (2010). An enhanced genetic algorithm with simulated annealing for job-shop scheduling. *International Journal of Engineering, Science and Technology, 2*(1), 144–151.

Van Laarhoven, P. J., Aarts, E. H., & Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations Research, 40*(1), 113–125. http://dx.doi.org/10.1287/opre.40.1.113.

Wang, L., & Zheng, D. Z. (2001). An effective hybrid optimization strategy for job-shop scheduling problems. *Computers & Operations Research, 28*(6), 585–596. http://dx.doi.org/10.1016/S0305-0548(99)00137-9.

Wang, X., & Duan, H. (2014). A hybrid biogeography-based optimization algorithm for job shop scheduling problem. *Computers & Industrial Engineering, 73*, 96–114. http://dx.doi.org/10.1016/j.cie.2014.04.006.

Watanabe, M., Ida, K., & Gen, M. (2005). A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. *Computers & Industrial Engineering, 48*(4), 743–752. http://dx.doi.org/10.1016/j.cie.2004.12.008.

Yamada, T., & Nakano, R. (1992). A genetic algorithm applicable to large-scale job-shop problems. In *Proceedings of the 2nd international workshop on parallel problem solving from nature (PPSN '92)* (pp. 281–290). Brussels, Belgium.

Yusof, R., Khalid, M., Hui, G. T., Md Yusof, S., & Othman, M. F. (2011). Solving job shop scheduling problem using a hybrid parallel micro genetic algorithm. *Applied Soft Computing, 11*(8), 5782–5792. http://dx.doi.org/10.1016/j.asoc.2011.01.046.

Zhang, C., Li, P., Guan, Z., & Rao, Y. (2007). A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research, 34*(11), 3229–3242. http://dx.doi.org/10.1016/j.cor.2005.12.002.

Zhou, H., Feng, Y., & Han, L. (2001). The hybrid heuristic genetic algorithm for job shop scheduling. *Computers & Industrial Engineering, 40*(3), 191–200. http://dx.doi.org/10.1016/S0360-8352(01)00017-1.

Zobolas, G. I., Tarantilis, C. D., & Ioannou, G. (2008). Exact, heuristic and meta-heuristic algorithms for solving job shop scheduling problems. In F. Xhafa & A. Abraham (Eds.), *Metaheuristics for scheduling in industrial and manufacturing applications* (pp. 1–40). Berlin: Springer.

Zobolas, G. I., Tarantilis, C. D., & Ioannou, G. (2009). A hybrid evolutionary algorithm for the job shop scheduling problem. *Journal of the Operational Research Society, 60*(2), 221–235. http://dx.doi.org/10.1057/palgrave.jors.2602534.