HYBRID EVOLUTIONARY APPROACH FOR MULTI-OBJECTIVE JOB-SHOP SCHEDULING PROBLEM

Kazi Shah Nawaz Ripon
Computer Science and Engineering Discipline,
Khulna University, Khulna 9208, Bangladesh.
Email: ripon@cseku.ac.bd

ABSTRACT

Over the years, various evolutionary approaches have been proposed in efforts to solve the Job-Shop Scheduling Problem (JSSP), a particularly hard combinatorial optimization problem. Unfortunately, most of these approaches are limited to a single objective only, and often fail to meet the requirements for real-world applications. Previously, we proposed several evolutionary approaches for multi-objective JSSP using the Jumping Genes Genetic Algorithm (JGGA) [1], [2]. Simulation results indicated that these approaches are capable of maintaining consistency and convergence of the trade-off, non-dominated solutions. In some rare cases, however, the solutions may be too diverse due to the additional diversity that occurs naturally from the jumping operations introduced in JGGA. This paper extends the idea by describing a hybrid approach that alleviates the difficulty outlined above. Experimental results reveal that our proposed hybrid approach can search for the nearly-optimal and non-dominated solutions with better convergence by optimizing multiple criteria simultaneously. Concurrently, it is capable of producing a set of controlled, diverse solutions that provide a wide range of alternative scheduling choices.

Keywords: Job-Shop Scheduling Problem (JSSP), Multi-Objective Evolutionary Optimization, Jumping Genes Genetic Algorithm (JGGA).

1.0 INTRODUCTION

Scheduling allocates shared resources over time to competing activities with hard or soft constraints given. In essence, scheduling can be considered as a searching or optimization problem, with the goal of finding the best schedule. Among the various types of scheduling problems, JSSP is one of the most challenging. Except for strongly restricted special cases, JSSP is an NP-hard problem that has also been considered the worst among combinatorial problems [3]. The exact methods, such as the branch and bound, as well as dynamic programming, are computationally expensive to use in searching for an optimum scheduling solution when the search space is large—which is natural in the present-day real-world of JSSP. Hence, it makes more sense to employ near-optimal solutions to overcome this difficulty. Stochastic search techniques, such as Evolutionary Algorithms (EAs), can be used to find such solutions.

Over the years, several classes of scheduling problems have been investigated; consequently, many different methods have been developed. However, most research on scheduling concerns with a single objective: the optimization of makespan (the maximum completion time of all jobs). Real-life scheduling problems, however, often require the decision maker to consider a number of criteria before arriving at a decision. A solution that is optimal with respect to one given criterion might be a poor candidate for some others like mean flow-time (the average response of the schedule to the individual demands of jobs for service), tardiness (the lateness of any job measures the conformity of the schedule to that job's committed date), etc. Therefore, the trade-offs involved in considering several different criteria provide useful insights for decision makers. Surprisingly, research in this important field has been scarce in comparison with the research in single-criterion scheduling. The goal of multi-objective JSSP is to find as many different schedules as possible that are near-optimal and non-dominated with regard to different objectives. In this work, we apply makespan and total tardiness as the objectives of our multi-objective scheduling approach.

The JGGA [4] is a recent Multi-Objective Evolutionary Algorithm (MOEA) that imitates a jumping gene phenomenon discovered by Nobel Laureate Barbara McClintock for her work on corn plants. In previous works [1], [2], we proposed several multi-objective evolutionary scheduling approaches based on JGGA for JSSP. In those approaches, the jumping gene operations exploit scheduling solutions around the chromosomes and the conventional genetic operators globally explore solutions from the population using multiple objective functions.

The central idea is to provide local search capability to fine-tune scheduling solutions during evolution. The JGGA is robust for searching the non-dominated solutions while considering both convergence and diversity [4], [5]. In scheduling, it is important to obtain converged and diverse solutions. Converged solutions guarantee that the most near-optimal Schedules considering multiple criteria can be obtained. Diverse solutions – in particular, the extreme solutions – are useful in helping the production manager to select the most compromised schedule from the non-dominated set of solutions according to specific objectives.

Experimental results of our previous approaches [1], [2] indicated that JGGA-based approaches are capable of producing near-optimal and non-dominated solutions. By following the natural tendency of the conventional JGGA, they can find extreme solutions. However, this characteristic causes some deficient effects – scheduling solutions to be too diverse in certain cases, and the number of non-dominated solutions per generation to be relatively small. This paper addresses this issue by proposing an improved version of the JGGA-based, multi-objective evolutionary scheduling approach that employs a hybrid method. Compared to other existing heuristic-based evolutionary scheduling approaches, the proposed approach can obtain overall good performance for all applied benchmark problems by finding non-dominated solutions with better convergence and diversity. The proposed hybrid approach is also compared to another well-established MOEA (NSGAII [6]) based multi-objective scheduling approach. Much better performance of the proposed approach has been noted.

The remainder of this paper is organized as follows: Section 2 describes scheduling, JSSP, and related works. The importance of multi-objective JSSP and a literature review of the multi-objective Genetic Algorithm (GA) for JSSP are presented in Section 3, as well as a discussion of the importance of hybridization. Section 4 offers a brief overview of the original JGGA and its effects on multi-objective JSSP. The detailed description of the proposed hybrid JGGA for solving JSSP is discussed in Section 5. Section 6 discusses the experimental results, and Section 7 outlines the conclusions of this paper.

2.0 SCHEDULING AND JOB-SHOP SCHEDULING PROBLEM

2.1 Scheduling Problem

The objective of scheduling is to efficiently allocate shared resources (machines, people etc) over time to competing activities (jobs, tasks, etc) such that a certain number of goals can be economically achieved and the given constraints can be satisfied. The solutions that satisfy these constraints are called feasible schedule. In general, the construction of a schedule is an optimization problem of arranging time, space, and (often limited) resources simultaneously. Hence, if scheduling is regarded as a search problem, it is desirable to search for any feasible schedule, and if it is considered to be an optimization problem, it is preferred to search for the best feasible schedule. In this paper, we focus on solving the JSSP since it is widely found in the industry and it is considered as a representation of many general scheduling problems in practice.

2.2 Job-Shop Scheduling Problem (JSSP)

A job-shop scheduling involves processing of the jobs on several machines without any 'series' routing structure. The $n \times m$ JSSP can be described by a set of n jobs $\{J_j\}_{1 \leq j \leq n}$ which is to be processed on a set of m machines $\{M_r\}_{1 \leq r \leq m}$. Each job has a technological sequence of machines to be processed. The processing of job J_j on machine M_r is called the operation O_{jr}. Operation O_{jr} requires an exclusive use of machine M_r for an uninterrupted duration P_{jr}, its processing time. A schedule is a set of completion times for each operation $\{C_{jr}\}_{1 \leq j \leq n, 1 \leq r \leq m}$ that satisfies given constraints. The challenge here is to determine the optimum sequence in which the jobs should be processed in order to optimize one or more performance measure, such as the makespan, the mean flow time, or the total tardiness of jobs etc.

2.3 Complexity of JSSP

The complexity of JSSP increases with its number of constraints and the size of search space. Except for some strongly restricted special cases, the JSSP is an NP-hard problem and finding an exact solution is computationally intractable [3]. Even a simple version of the standard JSSP is NP-hard if the performance measure is the makespan and $m > 2$. For the standard JSSP, the size of search space is $(n!)^m$ and it is computationally unfeasible to try every possible solution since the required computation time increases exponentially with the problem size. In practice, many real-world JSSPs have larger number of jobs and machines as well as additional constraints, which in turn further increase its complexity.

2.4 Related Works for Traditional and Heuristic Approaches

The JSSP has been extensively studied over the last forty years. A wide variety of approaches have been proposed in many diverse areas such as operations research, production management, and computer engineering. The comprehensive surveys of the general JSSPs are found in [7], [8]. Branch and Bound is an enumerative strategy where a dynamically constructed tree representing the solution space of all feasible schedules is implicitly searched. Although this method has proven to be very useful for small to medium sized problems, its excessive computing time prohibits its application to large problems [9]. In addition, their performance is quite sensitive to individual instances and initial upper bound values. Tabu Search (TS) has revealed to be an effective local search algorithm for the JSSP [10]. However, the best solution found by TS may depend on the initial solution used. This is essentially due to the fact that, like many other local search technique, TS starts its search from a single solution, which may lead the search to a dead-end despite the presence of the taboo mechanism. This happens especially when TS is applied to particularly hard optimization problem like JSSP. Simulated Annealing (SA) is the most popular technique in threshold algorithm category. It has been applied extensively to JSSP [11], [12] and can avoid local maxima/minima. However, as SA is a generic technique, it is unable to achieve good solutions quickly. Shifting Bottleneck (SB) [11] has had the greatest influence on approximation methods and was the first heuristic to solve the *mt10* problem. The primary weakness of this algorithm, however, is the high computing effort required and many re-optimizations are necessary to achieve these results. In addition, best solutions are achieved from several different parameter settings. Another fundamental problem is the difficulty in performing re-optimization and the generation of unfeasible solutions. Inspired by the principles of behavior found in real ant colonies, the Ant Colony Optimization (ACO) meta-heuristic has been applied to a variety of scheduling problems with promising results. However, the available results are quite poor and have yet to prove with currents state-of-art algorithms [13]. Greedy Randomized Adaptive Search Procedure (GRASP) is a problem-space-based method that consists of a constructive and an iterative phase. It generates many different starting solutions using fast problem-specific constructive procedures, which are then used by local search. Even though GRASP has been applied successfully to several NP-Complete problems, the limited results available so far for JSSP are quite poor [8].

2.5 Related Works for Genetic Algorithms

Genetic Algorithm (GA), proposed by Holland [14], has been successfully applied to solve many combinatorial optimization problems including scheduling. GA finds the best solution from many points of the search domain simultaneously rather than analyzing one point of the domain at a time and for that GA does not get stuck in local optima easily. In addition, GA is not only effective to perform global search, but also flexible to hybridize with other domain-dependent heuristics or local search techniques for solving specific problems. As the exact methods usually take exponential time and many heuristic approaches can only find sub-optimal solution for large JSSP, GA becomes a more popular approach to solve JSSP. It is already shown in the literature that the GA-based approaches can often achieve more robust and better performance than many traditional and heuristic approaches applied in JSSP [15].

Various kinds of GA-based solution methods have been proposed by a significant number of researchers so far. It is impossible to fairly compare these methods because they are very much focused on their own specific scheduling problems. Nakano et al. used a conventional GA with binary-chromosomes for the JSSP [15], but his representation did not guarantee to produce legal schedules, requiring a repair mechanism to correct illegal schedules after genetic operations. His repair process usually produces a fairly similar but legal schedule at the expense of significant computational cost. Fang et al. also successfully used the GA to solve the JSSP with a variant of ordinal representation that is developed for the validity of schedule under the conventional cut-and-mix crossover and mutation [16]. However, the genotype representation is redundant and thus false competition occurs. An in-depth survey of the applications of GAs in JSSP is found in [17].

3.0 IMPORTANCE OF MULTI-OBJECTIVE JOB-SHOP SCHEDULING

In many real-world JSSPs, it is necessary to optimize several criteria, such as the length of a schedule or the utilization of different resources simultaneously. In general, the minimization of makespan is used as the optimization criterion in a single-objective JSSP. However, the minimization of tardiness, flow time, machine idle time, etc. are also important criteria in JSSP. As discussed in [13], makespan may not be the only commercial interest in scheduling; other objectives are equally important. It is desirable, therefore, to generate many near-optimal schedules considering multiple objectives, according to the requirements of the production order or customer demand. On the other hand, if multiple objectives conflict with each other, the production manager need not omit any required objective before the assistance of a multi-objective scheduler. Based on the principle of multi-objective optimization, obtaining an optimal schedule solution that satisfies all the objective functions is nearly impossible. Accordingly, it is preferable to obtain as many different Pareto-optimal schedules as possible. These should be non-dominated, converged to, and diverse along the Pareto-optimal front, with respect to the multiple criteria.

3.1 Related Works of Genetic Algorithms for Multi-Objective JSSP

During the last decades a great deal of attention has been paid to solve JSSP with GAs. Although dealing with multiple objectives has received more and more attention over the last few years, these approaches are still considered limited, and mostly dominated by the unrealistic single-objective GA. Moreover, most of the multi-objective GA approaches for JSSP are mainly based on aggregation of preferences method, in which multiple objectives are combined into a single scalar objective using weighted coefficients. As the relative weights of the objectives are not exactly known in advance and cannot be pre-determined by the users, the objective function that has the largest variance value may dominate the multi-objective evaluation. As a result, inferior non-dominated solutions with poor diversity will be produced. Hence, it is essential to apply a *posteriori* articulation of preferences and present all the Pareto-optimal solutions to the decision makers in advance. At the time of this writing, there are only a few multi-objective GA based JSSP approach utilizing *posteriori* articulation of preferences are available in the literature [18].

3.2 Necessity for Hybrid Approach

Experimental results from our previous works [1], [2] demonstrate that JGGA-based scheduling approaches are capable of producing a set of non-dominated solutions close to the Pareto-optimal front and that have satisfactory diversity. In some rare cases, however, the solutions may be criticized for being too diverse. In fact, the properties exhibited by the solutions act upon the basic properties of JGGA. The jumping genes in JGGA could jump in and out of the chromosomes at different loci and thus, it could provide additional diversity [5]. This additional diversity is critical for multi-objective optimization, particularly during the early generations of the evolutionary process. However, once the population is somewhat converged in comparison to initial haphazard distribution, the jumping operations may produce some solutions that are too diverse. As observed in previous works [2], [5], JGGA can identify the extreme solutions, which are very useful for the human experts when selecting the best compromise schedule from a set of non-dominated solutions. Still, in some rare cases, the gap between the average and best values of the solutions produced by JGGA is more than that of the solutions produced by NSGAII. Indeed, the number of non-dominated solutions per generation is not satisfactory enough. Although it happens rarely, care should be taken to reduce these instances. We believe that a hybrid approach discussed in Section 5.5 can tackle this problem.

4.0 JUMPING GENES GENETIC ALGORITHM (JGGA)

To incorporate the jumping genes paradigm into an EA framework, a new operation — *cut and paste* or *copy and paste* is introduced after the selection process. The implementation of JGGA is that each chromosome has some consecutive genes which are selected as a transposon. The number of transposons in a chromosome can be greater than one and the length of each transposon can be more than one unit. The locations of the transposons are also assigned randomly, but their contents can be transferred within the same or even to different chromosomes in the population pool. The actual implementation of cut and paste operation is that the element is cut from the original site and pasted into a new site. In the case of copy and paste, the element replicates itself, with one copy of it inserted into a new site, while the original one remains unchanged at the same site. The non-dominated sorting strategy, crowding-distance mechanism, and elitism strategy used in JGGA are the same as used in NSGAII. For the detailed description of the JGGA, the reader may refer to [4], [5].

Every conventional genetic operator in GA employs only vertical transmission of genes from generation to generation. However, the jumping gene operators introduce a kind of horizontal transmission. The most important feature of JGGA is its capability to exploit local search heuristics by emulating a genetic phenomenon of horizontal transmission in which genes can jump from one position to another either within its own or to the other chromosomes under multiple stresses. Indeed, the jumping gene operations are better ways for exploration and exploitation than the use of Pareto-optimal solutions itself only. Therefore, it creates more chances to achieve better convergence and diversity, as well as to avoid premature convergence.

4.1 Advantages of JGGA for Multi-Objective JSSP

It is well known that GA is not very effective for fine-turning the solutions that are already close to the optimal solution as the crossover operators may not be sufficient enough to generate feasible schedules [19]. Hence, it is necessary to integrate some local search strategies in GA for enhancing the Pareto-optimal solutions. The rationale behind the hybridization is that GA is used to perform global exploration among the population, while local search strategy is used to perform local exploitation around the chromosomes. In addition, it should be noted that as the length of chromosome increases with the problem size of JSSP, the multi-objective GA may suffer from premature convergence in large search space [1]. To combat this, the jumping gene operations proposed in JGGA offer the local search capability to exploit solutions around the chromosomes, while the usual genetic operators globally explore solution from the population using multiple objective functions.

5.0 DESIGN ISSUES OF THE PROPOSED HYBRID APPROACH

5.1 Chromosome Representation

Representations used in GAs to solve JSSPs can be grouped into two basic encoding approaches [17]: direct and indirect. The direct approach encodes a schedule as a chromosome and the genetic operators are used to evolve these chromosomes into better schedules. This approach does not require any schedule builder. However, applying simple genetic operators on direct representation string often results in unfeasible schedule solutions. For this reason, domain-specific genetic operators are required. In indirect representation, the chromosome encodes a sequence of decision preferences, for example simple ordering of jobs in a machine or any heuristic rules, and a schedule builder is required to decode the chromosome into a schedule.

In our work, indirect representation incorporated with a schedule builder is applied. This chromosome representation is implemented with an un-partitioned operation-based representation where each job integer is repeated *m* times (*m* is the number of machines), and it is mathematically known as “permutation with repetition” [19]. By scanning the permutation from left to right, the *k*-th occurrence of a job number refers to the *k*-th operation in the technological sequence of this job as depicted in Fig. 1. In this representation, it is possible to avoid the schedule operations whose technological predecessors have not been scheduled yet. Therefore, any individual can be decoded into a feasible schedule, but two or more different individuals may be translated into an identical schedule. The advantage of such a scheme is that it requires a very simple schedule builder because all the generated schedules are legal.

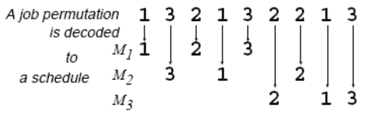


Fig: 1: Permutation with repetition approach for a 3X3 JSSP

5.2 Schedule Builder

A schedule builder transforms the chromosomes into a feasible schedule. The schedule builder is a module of the evaluation procedure and it should be chosen with respect to the performance-measure of optimization. Usually, the minimization of makespan plays the major role in converting the chromosomes into feasible schedule [20]. A schedule is called semi-active when no operation can be started earlier without altering the operation sequences of any machine. Very often, it is possible to reduce the makespan of a semi-active schedule by shifting an operation without delaying other jobs. When no such shifting can be applied to a schedule, it is called an active schedule. Since searching for active schedules brings a huge reduction of the search space and an optimal schedule is clearly active [11], it is safe and efficient to limit the search space to the set of all active schedules.

One of the efficient approaches to generate an active schedule builder is the Giffler & Thompson algorithm [21]. Here we employ a variant of hybrid Giffler & Thompson Algorithm proposed by Varela et al [22]. We modified this algorithm in order to fit with the representation of chromosome and produce active schedules only [1]. In the following algorithm, *S* is the schedule being constructed. The set *A* is used to hold the set of schedulable operations, where an operations *o* is said to be schedulable if it has not been scheduled yet.

Algorithm 1. Hybrid Giffler and Thompson

```
1. Set $S = \{\ \}$;
2. Let $A = \{o_{j1} \mid 1 \le j \le N\}$;
while $A \ne \emptyset$ do
    3. $\forall o_i \in A$ let $st(o_i)$ be the lowest starting time of $i$, if scheduled now;
    4. Let $o_k \in A$ such that $st(o_k) + du(o_k) \le st(o) + du(o),\ \forall o \in A$; where $du(o)$ is the processing time for
        operation $o$. (if two or more operations are tied, pick the leftmost operation in the chromosome);
    5. Set $M^*$ is the machine that is to process $o_k$;
    6. Let $B = \{ o \in A \mid$ it is to process on machine $M^*$ and $st(o) < st(o_k) + du(o_k)\}$;
    7. Let $o_t \in B$ such that $st(o_t) \le st(o),\ \forall o \in B$;
    8. Select $o^* \in B$ such that $o^*$ is the leftmost operation in the chromosome and add $o^*$ to $S$ with starting time
        $st(o^*)$;
    9. Let $A = A \backslash \{o^*\} \cup \{SUC(o^*)\}$; where $SUC(o)$ is the next operation to $o$ in its job if any exists;
end while
```

5.3 Jumping Operations

The chromosome representation in this hybrid JSSP is different from that of the conventional JGGA. As a result, the direct application of the original jumping operators may create an illegal schedule. This problem worsens in the case of copy and paste. Since the original chromosome representation requires that the number of jobs must be equal to the number of machines on which it will be processed, the resulting chromosome may produce an unfeasible schedule after the operation. As a result, some problem-specific jumping operators are required. In this paper, we classify the jumping operators based on the number of participating parent chromosomes. With one chromosome, the concept is relatively easy. Two gene positions are selected randomly. Then the same number (random) of consecutive genes are selected and their positions exchanged.

We follow the concept of "partial schedule exchange crossover" [23] to implement jumping operations between two different chromosomes. At first, partial schedules from both chromosomes are selected randomly. A partial schedule is identified with the same job in the first and last positions of the selected portion, with the restriction that the first gene of the second partial schedule must be the same as that of the first. Then, by exchanging these two partial schedules, we get two offspring. Usually, the partial schedules being exchanged contain a different number of genes, and the offspring may not include – or may have operations in excess of – those required for each job. Therefore, the offspring may be illegal. For this reason, repair work is required to convert the chromosomes into a legal schedule. Readers may refer to our previous publications [1], [2] for the elaborate explanations with described examples.

5.4 Crossover and Mutation Operation

In this approach, we apply the Generalized Order Crossover (GOX) operator [24] for performing the crossover, and we follow the concept of the 'job-pair exchange mutation operator' [23] for mutation. In mutation, two non-identical jobs are picked up randomly and then they exchange their positions.

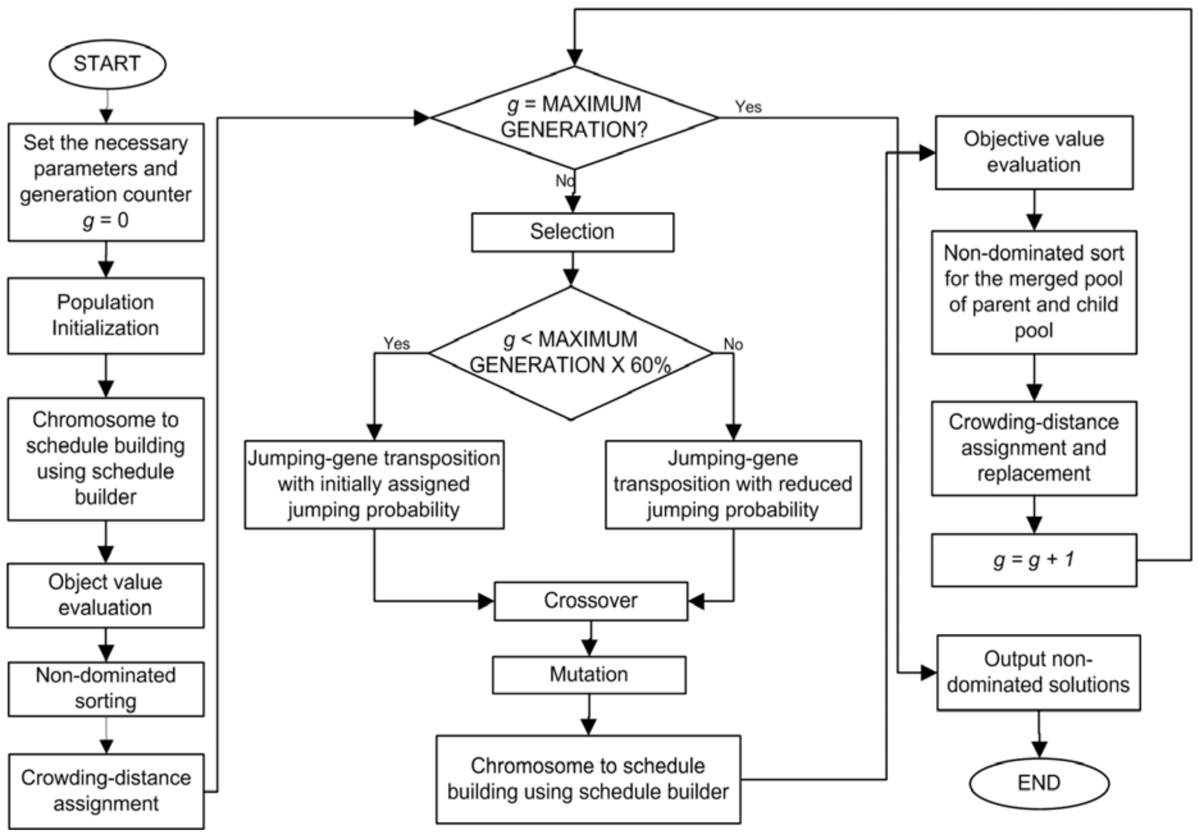

START
Set the necessary parameters and generation counter g = 0
Population Initialization
Chromosome to schedule building using schedule builder
Object value evaluation
Non-dominated sorting
Crowding-distance assignment
g = MAXIMUM GENERATION?
Yes
No
Selection
g < MAXIMUM GENERATION X 60%
Yes
No
Jumping-gene transposition with initially assigned jumping probability
Jumping-gene transposition with reduced jumping probability
Crossover
Mutation
Chromosome to schedule building using schedule builder
Objective value evaluation
Non-dominated sort for the merged pool of parent and child pool
Crowding-distance assignment and replacement
g = g + 1
Output non-dominated solutions
END

Fig. 2: Flowchart of hybrid JGGA for multi-objective JSSP

5.5 Hybrid approach

As stated early, the jumping genes can contribute a lot to attain the diversity while maintaining the appropriate convergence. In fact, the power and success of GA is mostly achieved by the diversity of the individuals of a population [25]. This is usually done by using the classical crossover and mutation operators. However, the genetic diversity of individuals can also be achieved more efficiently by several other mechanisms like gene insertion, duplication, or movement. With this respect, Mitchell and Forrest point out the importance of study other mechanisms for rearranging genetic material like jumping genes [26]. Conventionally, the influence of diversity is more important during the early generations than that of later [5]. For this reason, we utilize the full potential of the jumping operations during the early generations of the evolutionary process. We apply the full specified probability of the jumping operations in case of the first 60% cycles of the whole process. After that, the jumping probability is reduced to *1/3* of the full probability for the remaining cycles. The probabilities of crossover and mutation are kept the same for all generations. Experimental results justify that this hybrid approach produces more stable results than the previous approaches. Fig. 2 presents the complete evolutionary cycle of the proposed hybrid approach.

6.0 EXPERIMENTAL RESULTS

6.1 Benchmark Problems

To evaluate the proposed hybrid approach, we run the algorithm on various benchmark data. The first three well-known benchmark problems, known as *mt06, mt10* and *mt20*, are formulated by Muth and Thompson [27]. Applegate and Cook proposed a set of benchmark problems called the "ten tough problems", some of which still remain unsolved [28]. These problems consist of *abz7, abz8, abz9*, *la21, la24, la25, la27, la29, la38,* and *la40*. The problem data and the lower bound information are available in OR-library [29].

6.2 Experimental Evaluation and Discussions

First we perform the experiments in a single objective (makespan) context to justify its capability to optimize the makespan. After that, we show its performance as a multi-objective evolutionary JSSP algorithm by optimizing the makespan and the total tardiness, which are to be simultaneously minimized:

(i) Makespan = max[C_i] where C_i is the completion time of job *i*.

(ii) Total tardiness of jobs = $\sum_{i=1}^{n} \max[0, L_i]$ where L_i is the lateness of job *i*.

We also compare our proposed algorithm with another well-known MOEA (NSGAII [6]) based JSSP algorithm. For both algorithms, the experiments are conducted using 100 chromosomes and 150 generations. The probabilities of crossover and mutation are 0.9 and 0.3 respectively. For JGGA, the probability of jumping operations is initially 0.5. After evaluating 60% of generations (90 generations), the jumping probability has been reduced by 2/3 times of the initial value. Using the same settings, each benchmark problem is tested for thirty times with different seeds. Then, each of the final generation is combined and a non-dominated sorting [6] is performed to constitute the final non-dominated solutions.

6.3 Single Objective Context

The values provided in Table 1 show the makespan of the best schedules obtained in case of *mt* problems by some GA-based scheduling algorithms. The column labeled sGA is based on the GA using the simple mutation [30], where SGA is based on simple GA proposed by Nakano and Yamada [15]. LSGA and GTGA indicate GA-based JSSP algorithm incorporating local search [30] and GT crossover [11] respectively. From the Table 1, it can be easily found that the proposed JGGA based hybrid scheduling algorithm is capable of producing near-optimal values for the test problems. For *mt06* and *mt10* problems, it achieves the lower bound. In term of *mt20*, it cannot but it outperforms other contested algorithms.

Table 1: Makespan comparison with some GA based algorithms

Data	Lower Bound	sGA	GTGA	LSGA	SGA	JGGA
mt06	55	**55**	**55**	**55**	**55**	**55**
mt10	930	994	**930**	976	965	**930**
mt20	1165	1247	1184	1209	1215	**1180**

We also perform experiments to compare our proposed JSSP algorithm with some heuristic evolutionary approaches. Table 2 summarizes these results for the ten tough problems. As can be seen at the column headings, Nowi indicates Tabu Search based approach [10] and CBSA+SB indicates Simulated Annealing with Shifting Bottleneck heuristics [11]. Aart, Kopf, and Appl indicate Simulated Annealing results proposed in [12], GA performance in [31] and [28], respectively. The MSFX is based on Multi-Step Crossover Fusion [11]. The comparative results indicate that the proposed algorithm finds the near-optimal solution in case of five out of ten problems and optimal solutions can be found in three of them. However, other algorithms can also find the similar makespan in some problems, but not frequently like the proposed one. An exception is the performance of CBSA+SB. Despite that CBSA+SB is designed for single objective only, the main goal here is to find trade-off solutions for multi-objective JSSP. Moreover, for those cases where the proposed algorithm fails to achieve the known best results, it performs consistently and achieves very close to the near-optimal solutions.

Table 2: Makespan comparison with various evolutionary and heuristic methods

Data	**LD**	**Nowi**	**sGA**	**CBSA+SB**	**LSGA**	**Aarts**	**Kopf**	**Appl**	**MSFX**	**JGGA**
abz7	654	-	-	**665**	-	668	672	668	678	**665**
abz8	635	-	-	675	-	**670**	683	687	686	685
abz9	656	-	-	**686**	-	691	703	707	697	694
la21	1040	1047	1180	**1046**	1114	1053	1053	1053	**1046**	**1046**
la24	935	939	1077	**935**	1032	**935**	938	**935**	**935**	**935**
la25	977	**977**	1116	**977**	1047	983	**977**	**977**	**977**	**977**
la27	1235	1236	1469	**1235**	1350	1249	1236	1269	**1235**	**1235**
la29	1120	1160	1195	**1154**	1311	1185	1184	1195	1166	1156
la38	1184	**1196**	1435	1198	1362	1208	1201	1209	*1196*	1197
la40	1222	1229	1492	1228	1323	1225	1228	**1222**	1224	1225

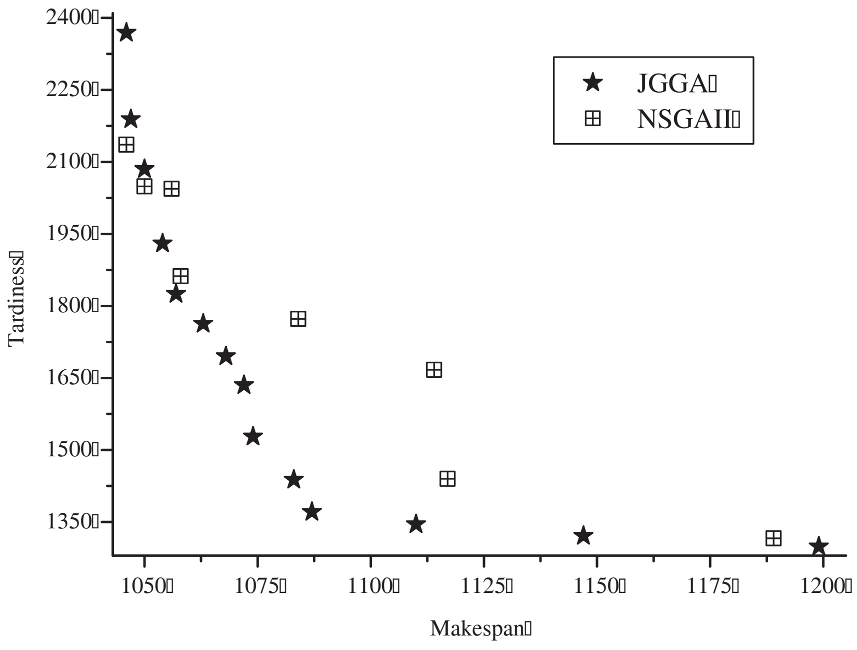
Table 3: Scheduling results for test problems

Data		**Makespan**		**Tardiness**		**Spread**
		Best	**Avg**	**Best**	**Avg**	**(*S*)**
mto6	JGGA	55	57.6	0	5.333	3.335
	NSGAII	55	57.48	0	4.357	3.639
mt10	JGGA	930	968.3	625	887.37	2.003
	NSGAII	930	966.5	630	893.25	2.046
mt20	JGGA	1180	1203.4	8359	8761.6	2.132
	NSGAII	1184	1233.5	7997	8961.16	3.335
abz7	JGGA	665	693.71	531	663.43	4.521
	NSGAII	667	691.19	553	674.28	8.377
abz8	JGGA	685	717.78	654	796.17	3.835
	NSGAII	686	723.66	757	818.63	7.25
abz9	JGGA	694	716.43	1009	1123	4.683
	NSGAII	690	720.2	1213	1340	4.50
la21	JGGA	1046	1082.64	1299	1699.5	3.362
	NSGAII	1046	1089.25	1316	1735.85	6.658
la24	JGGA	935	1035.08	1216	1406.38	1.812
	NSGAII	935	1036.62	1206	1372.14	3.066
la25	JGGA	977	999.94	1070	1221.5	1.104
	NSGAII	977	1016.75	1225	1505.5	3.019
la27	JGGA	1235	1271.87	3300	4117.12	2.478
	NSGAII	1235	1270.25	3436	4540.85	7.5013
la29	JGGA	1156	1184.13	4311	4603.32	6.794
	NSGAII	1160	1182.5	4419	4730.5	4.8333
la38	JGGA	1197	1232	1339	1461.63	2.716
	NSGAII	1196	1261.12	1343	1442.75	2.180
la40	JGGA	1225	1271	551	778.8	2.245
	NSGAII	1228	1272.57	535	807.28	4.225

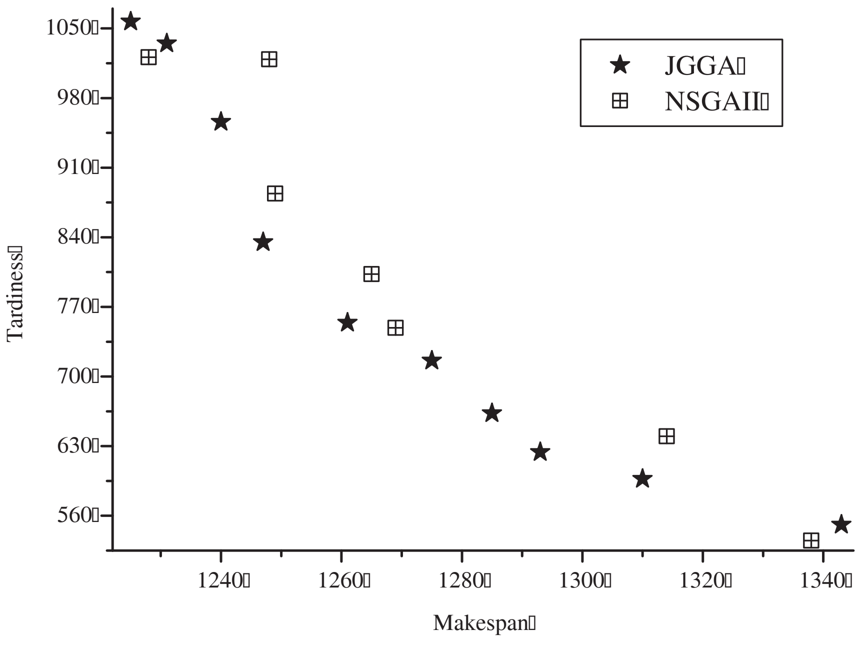
6.4 Multiple Objective Context

Multi-objective optimization deals with two goals – to find a set of solutions as close to the Pareto-optimal front as possible, and as diverse as possible. Table 3 shows the performance statistics of the evolutionary JSSP algorithms based on JGGA and NSGAII in the context of makespan and tardiness. The due dates for various benchmark problems are available in [2]. The results shown in the Table 3 indicate that both JGGA- and NSGAII-based algorithms perform well in achieving the near-optimal solutions. However, the hybrid approach clearly outperforms the other in terms of *mt10*, *abz7*, *abz8*, *la21*, *la25*, *la27*, and *la29*. Only in case of *la24* NSGAII outperforms JGGA. In other cases, the solutions produced by both algorithms are non-dominated to each other.

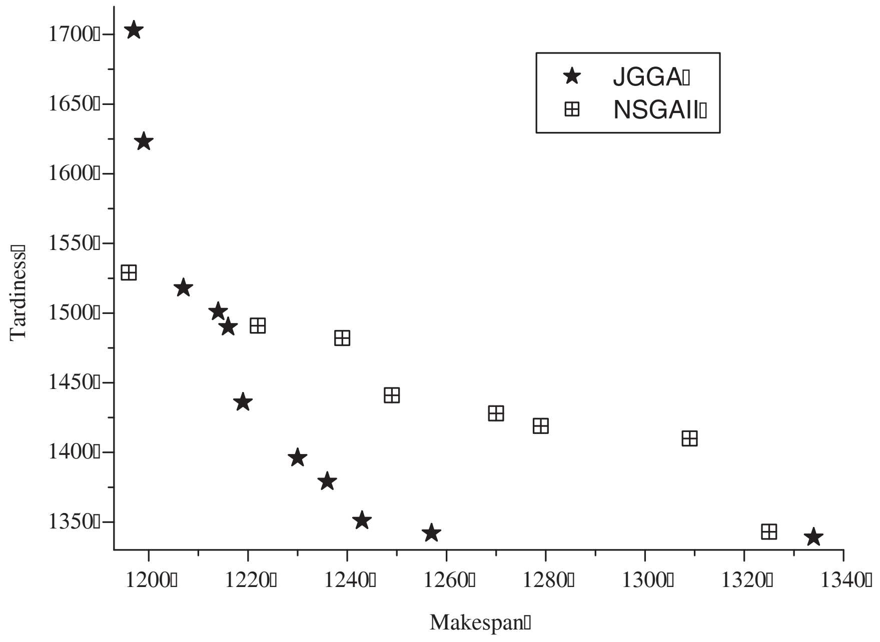
To illustrate the convergence and diversity of the solutions, the non-dominated solutions of the final generation produced by JGGA and NSGAII for the test problems *la21*, *la40* and *la38* are presented in Fig. 3. From these, it can be observed that the final solutions are well spread and converged. In particular, the solutions produced by our proposed hybrid approach are more spread than that of NSGAII; for this reason it is capable of finding extreme solutions. It can be further justified by the values of Space (*S*) metric [32] as specified in Table 3, given that an algorithm having a smaller *S* is better in terms of diversity.



(a) la21



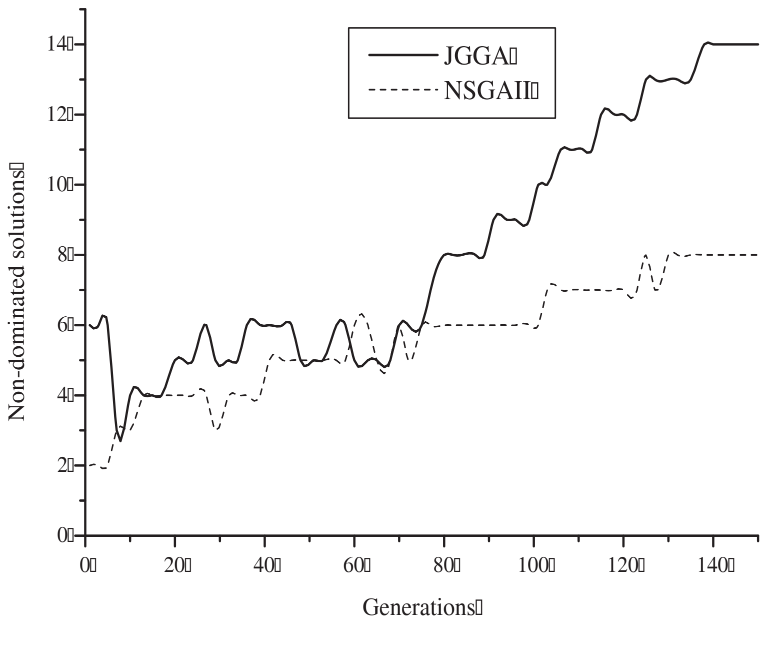
(b) la40



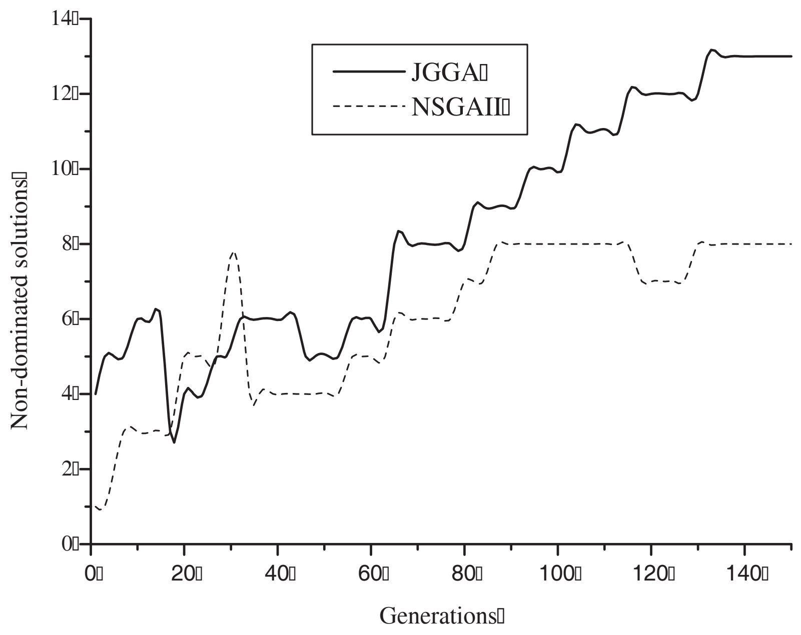
(c) la38

Fig. 3**:** Final Pareto-optimal front

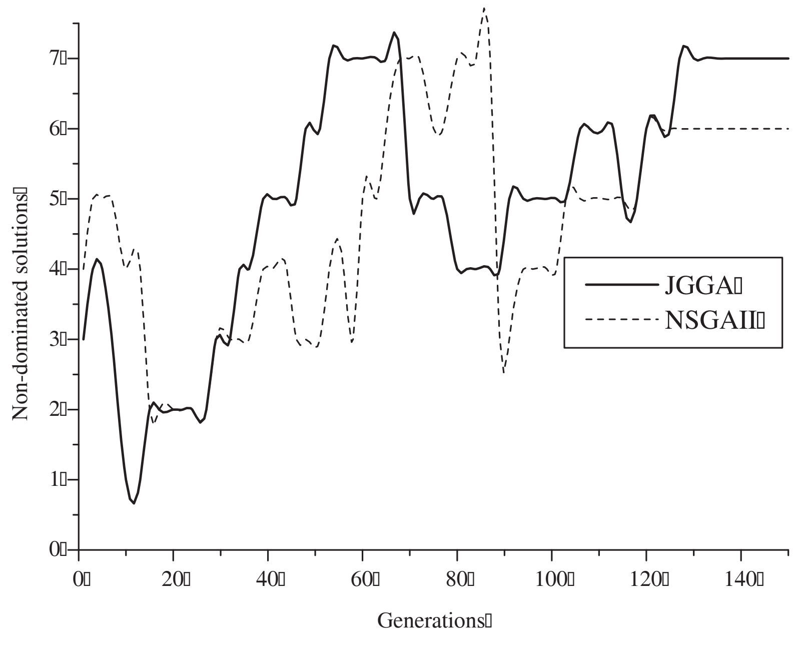
The properties exhibited by the solutions act upon the basic properties of JGGA. In general, the proposed hybrid approach is able to achieve consistent near-optimal solutions which are both well spread and converged. As compared to NSGAII, JGGA produces much more non-dominated solutions for most of the test problems. The plots of the obtained non-dominated solutions per generations in a single run for the test problems *la21*, *la24*, and *mt20* which is presented in Fig. 4 justify this. In fact, apart from *la25*, this approach produces more non-dominated solutions than NSGAII for the test problems.



(a) la21



(b) la24



(c) mt20

Fig. 4: Non-dominated solutions per generation (for a single run)

However, the ability of producing diverse solution has some side effects. From [1], [2], it can be found that in most cases the gap between the average and best values of the solutions produced by JGGA is more than that of the solutions produced by NSGAII. As discussed above, it is mainly for the extra diversity produced by the newly introduced jumping operations. On the contrary, diversity plays the most important role for producing well trade-off solutions [25], and the incorporation of jumping genes increases the genetic diversity in the population [4], [5]. To balance this phenomenon, we employ the hybrid approach in this work, and the results are very much promising.

From Table 4, we can find the gaps between the best and average values are much smaller than previous approach without hybridization. In our previous attempt [2], the values of tardiness were relatively unstable than makespan. However, the introduction of hybrid jumping operation makes it more stable. In fact, the best values for tardiness improve in some cases. Also, the best value, in terms of makespan, for *la38* improves than the previous approach, and the numbers of non-dominated solutions for most of the test problems increases. This trend justifies the application of the proposed hybrid approach in searching for a set of diverse and converged scheduling solutions. To summarize the result, the proposed hybrid multi-objective approach is capable of producing near-optimal and non-dominated solutions, which are also the lower bounds in many cases. The simulation results clearly show that our proposed hybrid approach is able to find a set of rational diverse solutions, which are also close to the Pareto-optimal front.

Table 4: Comparison between hybrid approach and without hybrid approach

Data		**Makespan**		**Tardiness**		**Spread (*S*)**
		Best	**Average**	**Best**	**Average**	
mto6	With hybrid	55	57.6	0	5.333	3.335
	Without hybrid	55	57.6	0	5.333	3.335
mt10	With hybrid	930	968.3	625	887.37	2.003
	Without hybrid	930	990	625	912.6	2.146
mt20	With hybrid	1180	1203.4	8359	8761.6	2.132
	Without hybrid	1180	1231.4	8359	8967.6	2.275
abz7	With hybrid	665	693.71	531	663.43	4.521
	Without hybrid	665	697.77	531	676.83	4.165
abz8	With hybrid	685	717.78	654	796.17	3.835
	Without hybrid	685	717.78	654	881.32	4.336
abz9	With hybrid	694	716.43	1009	1123	4.683
	Without hybrid	694	717.6	1009	1142	4.881
la21	With hybrid	1046	1082.64	1299	1699.5	3.362
	Without hybrid	1046	1085.75	1299	1746	6.658
la24	With hybrid	935	1035.08	1216	1406.38	1.812
	Without hybrid	935	1047.5	1249	1408.18	4.569
la25	With hybrid	977	999.94	1070	1221.5	1.104
	Without hybrid	977	998.83	1070	1320.5	1.133
la27	With hybrid	1235	1271.87	3300	4117.12	2.478
	Without hybrid	1235	1272.66	3300	4328.12	2.6778
la29	With hybrid	1156	1184.13	4311	4603.32	6.794
	Without hybrid	1156	1191.41	4349	4700.2	10.984
la38	With hybrid	1197	1232	1339	1461.63	2.716
	Without hybrid	1199	1241.88	1342	1519.22	2.180
la40	With hybrid	1225	1271	551	778.8	2.245
	Without hybrid	1225	1290.44	573	835.11	4.061

7.0 CONCLUSIONS

Most of the researches in scheduling generally concern with a single objective. Hence they are not suitable for real-world scheduling problems which are multi-objective by nature. JGGA is a recent MOEA which has been previously demonstrated as a robust evolutionary scheduling approach for solving multi-objective JSSP. However, it sometimes leads to the solutions which are too diverse – a limitation that can be overcome by the use of hybrid approach. The experimental results demonstrate that, as compared to other existing heuristic evolutionary scheduling approaches, the proposed hybrid approach can obtain overall superior performance. The comparative results with previous approach without hybridization reveal that the proposed hybrid approach performs well to balance the diversity versus convergence issue for multi-objective JSSP. The results also illustrates that the main strength of the proposed JGGA based hybrid approach is its ability to produce controlled-diverse solutions, while maintaining the consistency and convergence of the final trade-off non-dominated solutions.

REFERENCES

[1] K. S. N. Ripon, C. –H. Tsang and S. Kwong, "An Evolutionary Approach for Solving the Multi-Objective Job-Shop," *Studies in Computational Intelligence (SCI)*, Vol. 49, Springer-Verlag Berlin Heidelberg, 2007, pp. 165–195.

[2] K. S. N. Ripon, C. -H. Tsang and S. Kwong, "Multi-Objective Evolutionary Job-Shop Scheduling Using Jumping Genes Genetic Algorithm", in *Proceedings of the International Joint Conference on Neural Networks (ICJNN'06)*, Vancouver, Canada, July, 2006, pp. 3100-3107.

[3] M. Garey, D. Johnson and R. Sethi, "The Complexity of Flow Shop and Job Shop Scheduling*", Maths Ops Res*., Vol. 1, 1976, pp.117-129.

[4] T .M. Chan, K. F. Man, K. S. Tang and S. Kwong, "A Jumping Gene Algorithm for Multiobjective Resource Management in Wideband CDMA Systems", *The Computer Journal*, Vol. 48, No. 6, 2005, pp.749-768.

[5] K. S. N. Ripon, S. Kwong and K. F. Man, "A Real Coding Jumping Gene Genetic Algorithm (RJGGA) for Multiobjective Optimization", *Information Sciences*, Vol. 177, No. 2, 2007, pp. 632-654.

[6] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A Fast And Elitist Multiobjective Genetic Algorithm: NSGA-II", *IEEE Transaction on Evolutionary Computation*, Vol. 6, No.2, 2002, pp.182-197.

[7] J. Blazewicz, W. Domschke and E. Pesch, "The Job Shop Scheduling Problem: Conventional And New Solution Techniques", *European J. Operations Research*, Vol. 93, Vo. 1, 1996, pp. 1-33.

[8] A. S. Jain and S. Meeran, *A State-of-the-Art Review of Job-Shop Scheduling Techniques*, Technical Report, Department of Applied Physics, Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland, 1998.

[9] P. Bucker, B. Jurish and B. Sievers, "A Branch and Bound Algorithm for the Job Shop Scheduling Problem", *Discrete Applied Mathematics*, Vol. 49, 1994, pp.105-127.

[10] E. Nowicki and C. Smutnicki, "A Fast Tabu Search Algorithm for the Job-Shop Scheduling Problem", *Management Science*, 1996, Vol. 42, pp. 797–813.

[11] T. Yamada, *Studies on Meta Heuristics for Jobshop and Flowshop Scheduling Problems*, PhD. Thesis, Kyoto University, Japan, November, 2003.

[12] E. H. L. Aarts, P. J. M. van Laarhoven, J. K. Lenstra and N. L. J. Ulder, "A Computational Study of Local Search Algorithms for Job Shop Scheduling", *ORSA Journal on Computing*, Vol. 6, No. 2, 1994, pp. 118–125.

[13] E. Hart, P. Ross and D. Corne, "Evolutionary Scheduling: A Review", *Genetic Programming and Evolvable Machines*, 2005, Vol. 6, pp. 191-220.

[14] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[15] R. Nakano and T. Yamada, "Conventional Genetic Algorithm for Job-Shop Problems", in *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA '91)*, San Diego, CA, USA, 1991, pp. 474–479.

[16] H. L. Fang, P. Ross and D. Corne, "A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems", in *Proceedings of the Fifth International Conferences on Genetic Algorithms*, Urbana-Champaign, IL, USA, 1993, pp.375-382.

[17] R. Cheng, M. Gen and Y. Tsujimura, "A Tutorial Survey of Job-Shop Scheduling Problems Using Genetic Algorithms – I: Representation", *Computers and Industrial Engineering*, Vol. 30, No. 4, 1996, pp. 983-997.

[18] T. P. Bagchi, "Pareto-Optimal Solutions for Multi-Objective Production Scheduling Problems", in *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, Lecture Notes in Computer Science, Vol. 1993, Springer-Verlag, 2001, pp. 458-471.

[19] C. Bierwirth, "A Generalized Permutation Approach to Job Shop Scheduling With Genetic Algorithms", *OR Spektrum*, Vol. 17, 1995, pp. 87-92.

[20] T. Yamada and R. Nakano, "Scheduling by Genetic Local Search with Multi-Step Crossover," in, H.-M Voigt et al., (Eds.), *Lecture Notes in Computer Science*, vol. 1141, Springer, 1996, pp. 960-969.

[21] B. Giffler and G. Thompson, "Algorithms for Solving Production Scheduling Problems", *Operations Research*, Vol. 8, No 4, 1960, pp. 487-503.

[22] R. Varela, D. Serrano and M. Sierra, "New Codification Schemas For Scheduling With Genetic Algorithms", in J. Mira, and J. R. Álvarez, (Eds.), *Lecture Notes in Computer Sciences*, vol. 3562, Springer, 2005, pp. 11–20.

[23] M. Gen, Y. Tsujimura and E. Kubota, "Solving Job-Shop Scheduling Problem Using Genetic Algorithms", in *Proceedings of the Sixteenth International Conference on Computers and Industrial Engineering*, Ashikaga, Japan, 1994, pp. 576-579.

[24] C.Bierwirth, D. C. Matfield and H. Kopfer, "On Permutation Representation For Scheduling Problems", *Parallel Problem Solving from Nature*, Vol. 4, 1996, pp. 310-318.

[25] A. Simões and E. Costa, "Transposition: A Biologically Inspired Mechanism to Use with Genetic Algorithms", in *Proceedings of the Fourth International Conference on Neural Networks and Genetic Algorithms*, Portoroz, Slovenia, 1999, pp.178-186.

[26] M. Mitchell and S. Forrest, "Genetic Algorithms and Artificial Life," *Artificial Life*, Vol. 1, No. 3, 1994, pp. 267-289.

[27] H. Fisher and G. L. Thompson, "Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules" in J. F. Muth and G L. Thompson (Eds), *Industrial Scheduling*, Prentice Hall, 1963, pp. 225–251.

[28] D. Applegate and W. Cook, "A Computational Study of The Job-Shop Scheduling Problem", *ORSA Journal on Computing*, Vol. 3, No. 2, 1991, pp. 149–156.

[29] OR Library.
http://mscmga.ms.ic.ac.uk

[30] B. Ombuki and M. Ventresca, *Local Search Genetic Algorithms for Job Shop Scheduling Problem*, Technical Report No. CS-02-22, Brock Universoty, Canada, November 2002.

[31] D. C. Mattfeld, H. Kopfer and C. Bierwirth, "Control of Parallel Population Dynamics by Social-Like Behavior of GA-Individuals", *Lecture Notes in Computer Science*, Vol. 866, 1994, pp. 16-25.

[32] J. R. Schott, *Fault Tolerant Design Using Single and Multi-Criteria Genetic Algorithms*, Master's Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Boston, MA, 1995.

BIOGRAPHY

Kazi Shah Nawaz Ripon received his MPhil in Computer Science from City University of Hong Kong, Hong Kong, in 2006 and B.Sc (Engg.) in Computer Science & Engineering from Khulna University, Bangladesh, in 2000. Currently, he is an Assistant Professor in the Computer Science & Engineering Discipline, Khulna University, Bangladesh. He is also a member of Institute of Engineers, Bangladesh. His research areas include multi-objective optimization, evolutionary computation, and computational intelligence.