Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 48 (2012) 9280-9284

ISSN: 2229-712X

A Hybrid of genetic algorithm and simulated annealing for optimizing multi job shop scheduling

M. Nandhini[1], S.Kanmani[2], M. Thariga[1]

[1]Development Centre, Bharathiar University, Coimbatore,TamilNadu -46, India.

[2]Department of Information Technology, Pondicherry Engineering College, Puducherry-14, India.

[3]Department of Computer Science, Pondicherry University, Puducherry-14, India.

ARTICLE INFO

Article history:

Received: 2 May 2012;

Received in revised form:

15 June 2012;

Accepted: 3 July 2012;

Keywords

Multi Job-shop scheduling,
Heuristics,
Multi-objective optimization,
Roulette wheel selection,
Simulated Annealing.

ABSTRACT

A new hybrid approach with Genetic Algorithm and Simulated Annealing is proposed to solve Multi Job-Shop Scheduling problem, which is one of the well-known hardest combinatorial optimization problems. The main objective of multi job shop scheduling problem is to find a schedule of operations of each job in a set of jobs that can minimize the maximum completion time called makespan. To improve the makespan, SA algorithm has been designed and combined with genetic algorithm . Thus, hybrid GA is implemented over MJSSP and the effectiveness and efficiency is proved by getting promising results for different benchmark job-shop scheduling problem instances.

Introduction

Scheduling can be defined as a problem of finding an optimal sequence to execute a finite set of operations satisfying most of the constraints. The purpose of scheduling is to minimize the production time and costs. Production scheduling aims to maximize the efficiency of the operation and reduce costs. One of the most popular models in the area of scheduling is Multi Job-shop scheduling.

The Multi Job-shop scheduling problem (MJSSP) is an NP hard combinatorial optimization problem, which is very difficult to solve by conventional methods. The main goal of combinatorial optimization is finding the best possible solution from the set of feasible solutions. The amount of computation required to find optimal solution increases exponentially with problem size. The research on MJSSP not only promotes the development of relative algorithms in the field of artificial intelligence, but also provides means of solutions and applications for complex MJSSP.

The goal of MJSSP is to allocate machines to complete jobs over time, subject to the constraint that each machine can handle at most one job at a time. Thus, the MJSSP allocates resources over a specified time to perform a predetermined collection of tasks. The complexity of MJSSP increases with its number of constraints and size of search space. The constraints of the problem are so strong, making the valid search space of the problem too complicated.

Historically researchers have proposed several methods to solve the MJSSP using artificial intelligence methods. Exact methods are based predominantly on the Branch and Bound (BB) method, Dynamic Programming and Constraint Logic programming.

On the other hand, various heuristic and meta-heuristic algorithms, which are a quite good alternative such as simulated annealing (SA), tabu search (TS), genetic algorithm (GA), particle swarm optimization (PSO) ant colony optimization algorithm (ACO) and several other bio-inspired and nature-inspired algorithms (Zalinda et.al, 2004).

Nakano and Yamada (1991) proposed conventional genetic algorithms for job shop problems and they were among the first who applied a conventional GA that uses binary representation of solutions, to the job shop scheduling problem.

Yamada T and Nakano R(1992) proposed a genetic algorithm applicable to large-scale job-shop problems. In their work, they proposed a GA that uses problem-specific representation of solutions with crossover and mutation, which are based on the GT(Giffler–Thomson) algorithm.

José Fernando Gonçalves et.al.,(2002) proposed a hybrid genetic algorithm for the job shop scheduling problem. In their work, schedule was generated using a procedure that generates parameterized active schedules and at each schedule a local search heuristic was applied to improve the solution.

Juang (2004) proposed a hybrid of genetic algorithm and particle swarm optimization for recurrent network design. In this paper, the author described how the hybridization of GA with PSO overcomes each other's disadvantages.

Hong Zhou.et.al(2009) proposed a hybrid framework integrating a heuristic and a genetic algorithm (GA) for JSS to minimize weighted tardiness. In which, for each new generation of schedules, the GA determines the first operation of each machine, and the heuristic determines the assignment of the remaining operations.

Tamilarasi A and Anantha kumar T(2010) presented an enhanced genetic algorithm with simulated annealing for job shop scheduling problem. In their work, the best solution obtained without change for certain number of generations using GA is further improved by SA.

This paper gave an idea to hybridize the genetic algorithm with simulated annealing to solve the JSSP in different view

Tele:
E-mail addresses: mnandhini2005@yahoo.com

which is not attempted in the literature, in order to obtain optimum value with minimum computation time .

In this paper, a randomized constructive heuristic approach is employed to find the feasible schedule for MJSSP. Combining the advantages of GA and SA, a new hybrid approach is used to solve the MJSSP.

It is proposed that, to minimize the computation time, in each iteration, a few individuals that have a poor fitness value are substituted by a new solution generated by SA and the resulting set of solutions are moved to the solution space by GA.

This paper is organized as follows. In Section 2, a brief introduction of Multi job-shop scheduling and its constraints are presented. Section 3 describes the general description of GA and SA. The Hybrid GA and SA for solving MJSSP is presented in Section 4. Experimentally evaluated experimental results on a set of typical instances and comparative analysis are shown in Section 5. Conclusions and final remarks are discussed in Section 6.

Multi-Job Shop Scheduling Problem

Multi Job-shop scheduling problem (MJSSP) is considered as hard combinational optimization problem and it has been the subject of a significant amount of literature in the Operations Research (OR) areas. The MJSSP consists of n jobs and m machines. Each job must go through m machines to complete its work and it is considered that one job consists of m operations. Each operation uses one of m machines to complete one job's work for a fixed time interval. Once one operation is processed on a given machine, it cannot be interrupted before it finishes the job's work. The sequence of operations of one job should be predefined and may be different for any job. In general, one job being processed on one machine is considered as one operation noted as O_{ij} (means j^{th} job being processed on i^{th} machine, $1 \leq j \leq n, 1 \leq i \leq m$) . Each machine can process only one operation during the time interval.(Garey *et.al*., (1976) and Lawler *et.al*.,(1993).

The objective of MJSSP is to find an appropriate operation permutation for all jobs that can minimize the makespan Cmax i.e., to Minimize the final completion time.

The maximum completion time of the final operation in the schedule of n×m operations with minimum waiting time of jobs and machines is called makespan. The $n \times m$ JSSP, the problem can be modeled by a set of m machines, denoted by $M=\{1,2,\ldots,m\}$, to process a set of $n \times m$ operations, denoted by $O = \{1,2,\ldots,(n \times m)\}$

Where,

n : number of jobs

m : number of operations for one job

Oi : completed time of operation i (i=1,2,……($n \times m$))

ti : processing time of operation i on a given machine

$Cmax$: makespan

The problem can be understood with its known constraints (mandatory &optional) / assumptions as listed below.

- No machine can process more than one job at a time
- No job can be processed by more than one machine at a time
- The order in which a job visits different machines is predetermined by technological constraints
- Different jobs can run on different machines simultaneously
- At the moment T, any two operations of the same job cannot be processed at the same time
- Processing time on each machine is known
- Idle time of machines may be reduced
- Waiting time of jobs may be reduced

Thus, the MJSSP is to allocate machines to complete jobs over time, subject to the above constraint. The complexity of MJSSP increases with its number of constraints and size of search space. The constraints of the problem are so strong, making the valid search space of the problem too complicated.

Description of GA and SA

Genetic Algorithm

Genetic algorithm (GA) is a adaptive search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. GA belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

The basic concept of GAs is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest. As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem.

The Evolutionary process of GA is as follows: The evolution process starts from a population of random individuals. It is known as a generation. In each generation, the fitness of the whole population is evaluated, multiple individuals are stochastically selected from the current population based on their (fitness), modified (mutated or recombined) to form a new population(Sivanandham S N and Deepa S N ,2008), which becomes current population in the next iteration of the algorithm. GA is considered as one of the most powerful techniques in evolutionary algorithms, so GA has been employed as a tool that can handle a very complex search space with a high probability of success in finding the optimal solutions. The flow diagram for GA's evolutionary process is given in Fig.1.

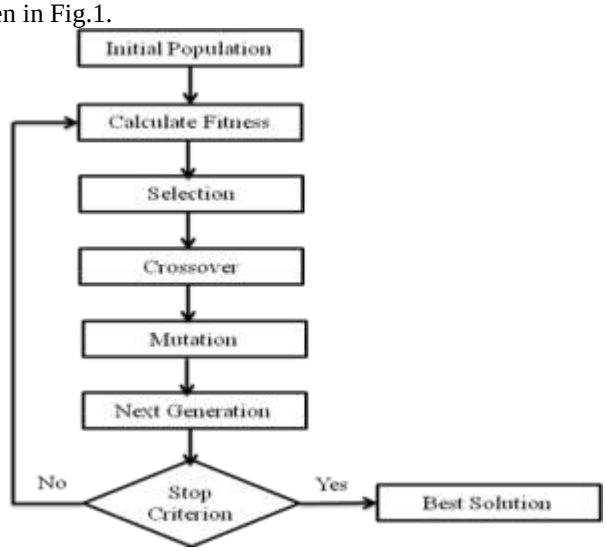

Fig. 1 Evolutionary process of GA

Simulated Annealing

Simulated annealing (SA) is a generic probabilistic metaheuristic for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space. It is often used when the search space is very large. The name and inspiration come from annealing in metallurgy.

SA is a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heat causes the atoms to become unstuck from

their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy; the slow cooling gives them more chances of finding configurations with lower internal energy than the initial one.

The Evolutionary process of SA is as follows: The evolution process starts from a random individual. Each step of the SA algorithm attempts to replace the current solution by a random solution. The new solution may then be accepted with a probability that depends both on the difference between the corresponding function values and also on a global parameter *T* (called the *temperature*), that is gradually decreased during the process. The dependency is such that the choice between the previous and current solution is almost random when *T* is large, but increasingly selects the better or "downhill" solution (for a minimization problem) as *T* goes to zero. The flow diagram for SA's iterative process is given in Fig.2.

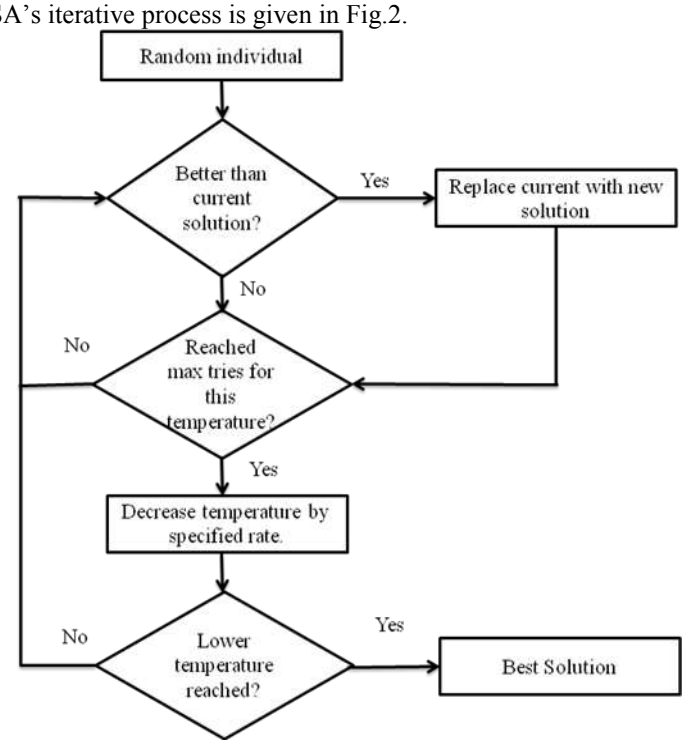

Fig. 2 Iterative process of SA

METHODOLOGY OF PROPOSED SYSTEM

Many algorithms have been proposed by hybridizing of GA with SA (Xia Weijun et. al.,2004, Sohrab Khanmohammadi and Hamed Kharrati, 2010, Elnaz Baghal Azardoost and Nrges Imanipour,2011).In order to reduce the computation time, and to reduce machine and jobs idle times, new approach is applied on mutation in the proposed work. Also, after mutation is applied over a schedule, to improve the fitness value SA is applied. The proposed architecture is shown in a schematic diagram in Fig. 4.

Initially, we start with a possible solution (chromosome) for the schedules which are randomly generated. This process is repeated to get 600 chromosomes. This set of chromosomes is our initial search space called population. The chromosomes evolve through successive iterations, called generations. During each generation, the chromosomes are evaluated, using some measures of fitness.

The state representation, fitness function and other operators that are used in our implementation are discussed in this section.

State Representation

State formation takes number of jobs, number of operations for each job, sequence of operations of each jobs, processing time of each job's operation and allotment of operation over machines as inputs. In our work, each state is represented as an array of structures as in Fig. 3. Each structure consists of job name and its operation as members.

Job No	Oper No	Job No	Oper No	.....	Job No	Oper No

Fig. 3 State Representation of MJSSP

For 3 Jobs and 3 Operations, the schedule is represented as follows in which the Operation sequences should not change:

O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}

O_{11} -> denotes first job's first operation

O_{23} -> denotes second job's third operation

Since, 3 jobs x 3operations, the array structure should be of 9 positions with all jobs operations.

Feasible Solution Generation

A Feasible schedule is generated by the Random constructive heuristics process. First operation in a schedule can be scheduled if it is been first operation of any of the jobs. Following this, unscheduled operations of remaining jobs are scheduled by verifying operation consistencies and capacity constraints. In the following example, the heuristic process is clearly mentioned. The operation sequence of any job should not change in the schedule. It should start from 1 and ends with m (no. of operations). An example of feasible solution is given in Fig.5.

O_{21}	O_{31}	O_{11}	O_{12}	O_{32}	O_{33}	O_{22}	O_{13}	O_{23}

Fig. 5 Feasible Schedule

O_{11} -> denotes first job's first operation

O_{23} -> denotes second job's third operation

Since, 3 Jobs x 3 Operations, the array structure should be of 9 positions with all jobs operations.

For n jobs on m machines MJSSP, the schedule is created in n x m dimensions.

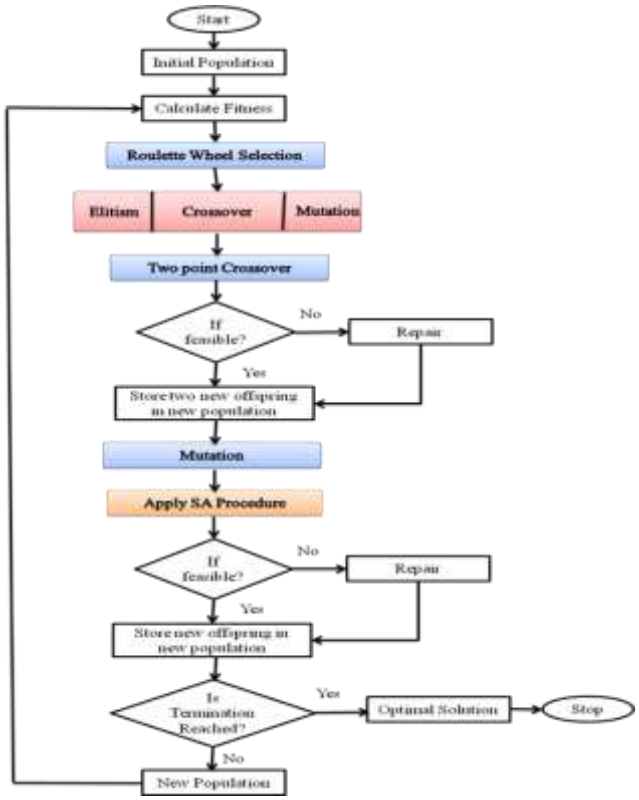

Fig. 4 Schematic diagram of proposed GA with SA

Fitness Function Evaluation

The evaluation of individual in the population is commonly regarded as the most computational demanding step. This step is executed in every generation for all the individuals. The evaluation of an individual can be done in isolation from the rest of the population, and no communication with the other individuals of the population is required or desired. The identification of good or bad solutions in a population is usually accomplished according to a solution's fitness. The essential idea is that a solution having a better fitness must have a higher probability of selection.

Min Z = Min (Makespan)

Where,

Job No. *I : 1..n*

Operation No. *J : 1..m*

In this work, the fitness function is a kind of objective function that quantifies the optimality of a solution, so that the particle may be ranked against other particle in the population. In our case, the fitness for the MJSSP is evaluated using Gantt chart. From the Gantt chart, the processing time is calculated.

Elitism Strategy

Elitism is a method, which copies few best chromosomes into new population. The rest is done in classical way. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution. Here, 10% of chromosomes having higher fitness values are copied into new population in order to retain the best solution in the next generation.

Roulette Wheel Selection

The basic part of the selection process is to stochastically select from one generation to create the offspring of the next generation. The requirement is that the fittest individuals have a greater chance of survival than weaker ones. This replicates nature in that fitter individuals will tend to have a better probability of survival and will go forward to form the mating pool for the next generation.

In roulette wheel selection, individuals are given a probability of being selected that is directly proportionate to their fitness. Two individuals are then chosen randomly based on these probabilities and produce offspring. Pseudo-code for a roulette wheel selection algorithm is shown in Fig. 6.

```
for all members of population
   sum += fitness of this individual
end for
   for all members of population
   probability = sum of probabilities + (fitness / sum)
   sum of probabilities += probability
end for

loop until new population is full
    do this twice
       number = Random between 0 and 1
      for all members of population
         if number > probability but less than next
probability
            then the current member is selected
      end for
    end
    create offspring
end loop
```

Fig .6. Pseudo Code for Roulette Wheel Selection

Two-Point Crossover

Crossover operator aims to interchange the information and genes between chromosomes. Therefore, crossover operator combines two or more parents to reproduce new children, then, one of these children may hopefully collect all good features that exist in its parents.[12]

In two-point crossover, two crossover points are chosen and the contents between these points are exchanged between two mated parents to produce two child offspring. The schematic representation of two-point crossover in MJSSP is shown in Fig. 7.

Mutation

Mutation means randomly deriving change to the gene sequence of the chromosomes. In GA, mutation is a purely random operator, in which the probability that a gene will mutate is of low value at the time of initialization.

In this paper, the procedure of mutation for MJSSP is proposed as follows:

Fig. 7 Two Point Crossover

(a) The distance of each job last operation and its previous operation is calculated

(b) From the distance set, a Job's last operation with larger distance is swapped next to its previous operation

(c) All jobs' operations after the swapped job operation is shifted one position to its right

The schematic representation of mutation is shown in Fig. 8.

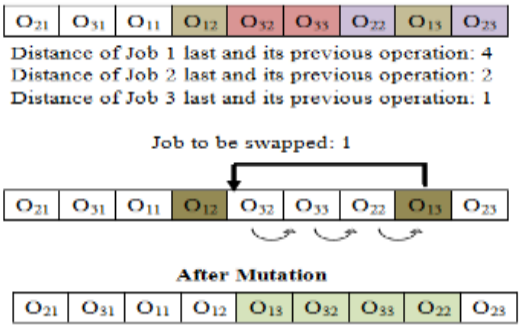

Fig. 8 Mutation

Repair

Repairing is mainly done for removing the violation of constraints after reproduction operation. This function has composed of two distinct tasks: *fault detection* and *fault correction.*

Knowing the location of the offending timeslots, repairing replaces these timeslots with free slots at first. If conflict rises, iteratively replaces with other timeslots entries in order to get rid of constraints violation.

Simulated Annealing

Simulated Annealing inserted within genetic algorithm is considered as an effective way to produce high quality solution than using stand alone genetic algorithm.

Especially after mutation in the GA process, we propose to apply simulated annealing to improve the candidate solution's fitness value. As its outcome, when applying SA, it produces the fitness value lower than the parent fitness value.

The feasibility of the schedule in each generation would be improved by hybridizing GA with SA. So that ,the optimal solution will be obtained by passing minimum number of generations. Hence, the convergence speed becomes high.

Termination Criterion

This iterative process continues until optimum fitness value is obtained.

COMPUTATIONAL RESULTS

We programmed the algorithms in Java Programming Language and run it on the Intel Pentium Core 2 Duo 3GHz Processor and 2 GB RAM configuration system with Windows XP as the platform.

To illustrate the performance of proposed algorithm , the benchmark instances of Lawrence(1984) with different sizes have been selected from the OR-Library (Beasley J E ,1990).

Evolution process started with 600 schedules in the initial population. If the optimum solution is reached , the process is terminated and the optimal solution is declared.

The optimal solutions obtained for some benchmark instances on different algorithms such as simple GA, GA with SA by [13] and proposed GA with SA are listed in the Table. 1 along with their CPU time. The CPU time is the time spent by CPU on each benchmark problem till receives the optimal schedule with the above machine configuration.

It is observed from Table.1 that the computation time of large size instances are lesser than the time taken by GA with SA[13] taken for comparison.

Hence, our proposed GA with SA can be used to solve MJSSP with multi objectives with lesser time is proved.

CONCLUSION

A hybrid algorithm with GA and SA helps in getting optimal solution with less computation time than existing algorithm is proved with its performance. It is suitable solivng large sized instances of multi objective combinatorial problems with more soft constraints like MJSSP. Hence , the superior results indicate the successful incorporation of GA and SA.

As future enhancement to this project, we have planned to hybrid GA with any of the metaheuristics algorithms to study the impact of natural inspiration over bioinspired process.

REFERENCES

[1] Beasley J E (1990) OR-library: Distributing test problems by electronic mail. J. of the Operational Research Society, 41(11),1069–1072.

[2] Elnaz Baghal Azardoost and Nrges Imanipour (2011) A Hybrid Algorithm for Multi Objective Flexible Job Shop Scheduling Problem. In Proc of the Int. Conf. on Industrial Engineering and Operations Management,795-801.

[3] Garey J E ,Johnson D S, and Sethi R (1976) The complexity of flow shop and job shop scheduling. Mathematics of Operations Research, 1(2),117–129.

[4] Hong Zhou, Waiman Cheung and Lawrence C. Leung (2009) Minimizing Weighted Tardiness of Job-shop Scheduling. Euro. J. of Operational Res., 194(3), 637-649.

[5] José Fernando Gonçalves, Jorge José de Magalhaés, Rua Dr,Roberto Frias, Jorge Jose, Magalhâes Mendes and Maurício G C Resende (2002) A Hybrid Genetic Algorithm for Job Shop Scheduling Problem. Euro. J. Operational Research, 167.

[6] Juang C F (2004)A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE

[7] Transactions on Evol. Computation, 34(2) ,997-1006.

[8] Lawler E L, Lenstra J K ,Rinnooy Kan A H G, and Shmoys D B (1993) Sequencing and scheduling: Algorithms and complexity. Handbooks in Operations Research and Management Science,4,445-552.

[9] Nakano R and Yamada T (1991) Conventional genetic algorithm for job shop problems., In Proc. of International Conference on Genetic Algorithms, 474–479.

[10] Ono, Yamamura M, and Kobayashi S (1996) A genetic algorithm for job-shop scheduling problems using job-based order crossover. In Proc. of ICEC '96, 547-552.

[11] Sivanandham S. N and Deepa S N(2008) Introduction to Genetic Algorithm :Springer.

[12] Sohrab Khanmohammadi and Hamed Kharrati(2010) A New Hybrid evolutionary Algorithm for Job-shop Scheduling Problems. Applied Mathematics and Informatics,51-56.

[13] Tamilarasi A and Anantha kumar T (2010) An enhanced genetic algorithm with simulated annealing for job-shop scheduling. International J. of Engn.Sci. and Technol. 2(1),144-151.

[14] Xia Weijun, Wu Zhiming, Zhang Wei and Yang Genke(2004) .A New Hybrid Optimization Algorithm for the Job-shop Scheduling Probem. In Proc. of the American Control Conference Boston, 5552-5557.

[15] Yamada T and Nakano R (1992) A genetic algorithm applicable to large-scale job-shop problems. In Proc. of The Second Int. Conf. on Parallel Problem Solving from Nature PPSN '92, 281-290.

[16] Zalinda Othman, Khairanum Subar and Norhashimah morad(2004) Job shop scheduling with alternative machines Using genetic algorithms. Journal Teknologi, 41(D), 67–78.

TABLE 1. COMPUTATIONAL RESULTS

Sl. No	Instance Name	Instance Size	Best Known Value	OPTIMAL VALUES			CPU TIME	
				Simple GA	GA with SA [13]	GA with SA (Proposed)	GA with SA (Proposed) (In Seconds)	GA with SA (In Seconds) [13]
1	LA01	10 x 5	666	794	666	666	14	15
2	LA02	10 x 5	655	686	655	655	182	190
3	LA03	10 x 5	597	666	597	597	28	36
4	LA04	10 x 5	590	620	590	590	45	48
5	LAO5	10 x 5	593	593	593	593	39	36
6	LA06	15 x 5	926	926	926	926	40	38
7	LA07	15 x 5	890	962	890	890	32	35
8	LA08	15 x 5	863	963	863	863	79	98
9	LA09	15 x 5	951	951	951	951	89	97
10	LA10	15 x 5	958	1011	958	958	90	95
11	LA16	10 x10	945	1008	945	945	80	94
12	LA17	10 x10	784	809	784	784	76	96
13	LA18	10 x10	848	916	848	848	198	230
14	LA19	10 x10	842	863	842	842	210	213
15	LA20	10 x10	902	928	902	902	157	170