IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 6, Ver. I (Nov – Dec. 2015), PP 16-24
www.iosrjournals.org

A Hybrid Evolutionary Optimization Model for Solving Job Shop Scheduling Problem using GA and SA

Dr. S. Jayasankari
Assistant Professor, Department of Computer Science,
P.K.R. Arts College for Women, Gobichettipalayam, Tamilnadu, India
Email id: jai.nivi.ravi@gmail.com

Abstract: *The heuristic optimization techniques were commonly used in solving several optimization problems. The present work aims to develop a hybrid algorithm to solve the scheduling optimization problem of JSSP. There are different variants of these algorithms that were addressed in several previous works. The impacts of these two kinds (Genetic Algorithm (GA) and Simulated Annealing (SA) based optimization model) of initial condition on the performance of these two algorithms were studied using the convergence curve and the achieved makespan. Even though genetic algorithm performed better than other evolutionary algorithms, it has some weakness. During running GA, sometimes, it will produce same result without any improvement. SA has a mechanism to overcome from that situation. During SA, if same result will be repeated, then it is rapidly changing the change in temperature variable and re-initiates another random search. By using this feature of SA, it has been implemented a hybrid based evolutionary model for solving JSSP by improving GA. Comparison has been made with the performance of the proposed SA-GA-Hybrid model with GA as well as SA.*
Keywords: *Job Shop Scheduling Problem, Optimization, Genetic Algorithm, Simulated Annealing.*

I. Introduction

Scheduling is an optimization method to make the best possible way to use the limited resources by making a suitable allocation of the constraint resources over a period of time. Those problems are combinatorial optimization problems, where a set of possible solutions is distinct or can be reduced to a discrete one, and the goal is to find the best possible solution. Scheduling falls under the NP-hard class of combinatorial optimization problems category. Developing of job processing schedules in a scheduling environment is a very difficult task. The number of possible schedules increases with the increase in the number of jobs and related operations. This growth makes it practically impossible to use mathematical programming or thorough search-based approaches to find the global optimum schedules.

In the recent competitive atmosphere in manufacturing and service industries, the operational sequencing and scheduling has become a crucial for survival in the marketplace. Companies have to produce their product advance as opposed to due date. Otherwise, it will affect upon reputation of a business. At the same time, the activities and operations need to be scheduled with the intention that the available resources will be used in an efficient manner. As a result, there is a great good scheduling algorithm and heuristics are invented. Most of the principal practical scheduling problems exist in stochastic and dynamic environment.

Stochastic is a problem in that some of the variables are undefined while dynamic problem is when jobs arrive randomly. In another way, the problems with ready time is known and fixed are known problems such as static and for problem where all the parameter such as processing times are known and fixed is called deterministic problems. Regardless of this, it is impossible to predict exactly when jobs will become available for processing.

The core objective in solving the job shop scheduling problem is to find the sequence for each operation on each machine that optimizes the objective function. The objective function that has been used in scheduling the job shop problem is minimization of makespan value or the time to complete all jobs.

1.1 Problem Definition

The Job Shop Scheduling Problem (JSSP) may be described as follows: Given 'n' jobs, each composed of several operations that must be processed on 'm' machines. Each operation uses one of the 'm' machines for a fixed duration. Each machine can process at most one operation at a time and once an operation initiates processing on a given machine it must complete processing on that machine without interruption. The operations of a given job have to be processed in a given order. The problem consists in finding a schedule of the operations on the machines, taking into account the precedence constraints, that minimizes the makespan (C_{max}), that is, the finish time of the last operation completed in the schedule.
The focus on job-shop scheduling problems composed of the following elements:

➢ **Jobs:** J = {J_1 , · · · , J_n } is a set of n jobs to be scheduled. Each job J_i consists of a predetermined sequence of operations. $O_{i,j}$ is the operation j of J_i. All jobs are released at time 0.

➢ **Machines:** M = {M_1 , · · · , M_m } is a set of m machines. Each machine can process only one operation at a time. And each operation can be processed without interruption during its performance on one of the set of machines. All machines are available at time 0.

➢ **Constraints:** The constraints are rules that limit the possible assignments of the operations. They can be divided mainly into following situations:

- Each operation can be processed by only one machine at a time (disjunctive constraint).
- Each operation, which has started, runs to completion (non-preemption condition).
- Each machine performs operations one after another (capacity constraint).
- Although there are no precedence constraints among operations of different jobs, the predetermined sequence of operation for each job forces each operation to be scheduled after all predecessor operations (precedence/conjunctive constraint).
- The machine constraints emphasize the operations can be processed only by the machine from the given set (resource constraint).

➢ **Objective:** The objective is to find a schedule that has minimum time required to complete all operations (minimum makespan).

1.2 Different Approaches for Solving Scheduling Problems

Job shop scheduling problem is a NP-hard problem with no easy solution. Branch-and-bound, Tabu search, and biologically inspired approaches such as GA, Swarm Intelligence and other stochastic model such as Simulated Annealing algorithm were proposed for achieving possible solutions to complex problems such as job shop scheduling.

SA is a stochastic heuristic algorithm which is used to resolve combinatorial optimization problems. Simulated Annealing optimization is analogous to the annealing of metals. This Simulated Annealing is different from other algorithms, such that SA uses a probability mechanism to have power over the process of jumping out of the local minimum. In the process of searching, SA is not only accepting better solutions, but also accepting worse solutions randomly with a certain probability. At high temperatures, the probability of accepting better solutions is comparatively big. With the decrease of the temperature, the chance of accepting worse solutions also move downs, and when the temperature closes in upon zero, SA no longer accepts any worse solution. These make SA have more chance to avoid getting trapped in a local minimum and avoid the limitation of other local search algorithms and the gradient algorithms. Because of its qualities above, SA has become an enormously accepted method for solving large-sized and practical problems like job shop scheduling, timetabling, traveling salesman and packing problem. However, like many other search algorithms, SA may get trapped in a local minimum or take a long time to find a reasonable solution. For these reasons, SA is often used as a part of a hybrid method.

II. The JSSP And The Hybrid Evolutionary Model For Solving JSSP

Let J = {0, 1, …, n, n+1} denote the set of operations to be scheduled and M = {1,..., m} the set of machines. The operations 0 and n+1 are dummy, have no duration and represent the initial and final operations. The operations are interrelated by two kinds of constraints. First, the precedence constraints, which force each operation j to be scheduled after all predecessor operations, P_j, are completed. Second, operation j can only be scheduled if the machine it requires is idle. Further, let d_j denote the (fixed) duration (processing time) of operation j. Let F_j represent the finish time of operation j. A schedule can be represented by a vector of finish times (F_1, , F_m, ... , F_{n+1}). Let A(t) be the set of operations being processed at time t, and let $r_{j,m}$ = 1 if operation j requires machine m to be processed and $r_{j,m}$ = 0 otherwise.
The conceptual model of the JSP can be described the following way (Goncalvesetal 2005):

Minimize F_{n+1} (C_{max}) …….(1)

Subject to:

$F_k \leq F_j - d_j$ j-1,…,n+2 ; $k \in P_j$ …….(2)

$\sum_{jA(t)} r_{j,m} \leq 1$ $m \in M$; $t \geq 0$ …….(3)

$F_j \geq 0$ j=1,….,n+1 …….(4)

The objective function (1) minimizes the finish time of operation n+1 (the last operation), and therefore minimizes the makespan. Constraints (2) impose the precedence relations between operations and constraints (3) state that one machine can only process one operation at a time. Finally (4) forces the finish times to be non- negative. The JSP is amongst the hardest combinatorial optimization problems. The JSP is NP-hard (Lenstra and Rinnooy Kan, 1979), and has also proven to be computationally challenging.

2.1 Simulated Annealing

Simulated Annealing (SA) is motivated by an analogy to annealing in solids. The idea of SA comes from a paper published by Metropolis etc al in 1953 [Metropolis, 1953]. The algorithm in this paper simulated the cooling of material in a heat bath. This is a process known as annealing. If a solid is heated up to the melting point of that metal and then cool it, the structural properties of the solid depend on the rate of cooling. If the liquid metal is cooled very slowly, large crystals will be formed. However, if the liquid is cooled quickly the crystals will contain imperfections. Metropolis's algorithm simulated the material as a system of particles. This algorithm simulates the cooling process by slowly lowering the temperature of the system until it converges to a steady, frozen state. In 1982, Kirkpatrick et al (Kirkpatrick, 1983) took the idea of the Metropolis algorithm and applied it to optimization problems. The idea is to use simulated annealing to search for feasible solutions and converge to an optimal solution.

Simulated annealing refers to the annealing process done on a computer by simulation. In this model, a parameter T, equivalent to temperature in annealing, is reduced slowly. The law of thermodynamics state that at temperature, t, the probability of an increase in energy of magnitude, δE, is given by (Kirkpatrick et al 1983):

$P(\delta E) = \exp(-\delta E / kt)$

Where k is a constant known Boltzmann's constant.

The simulation in the Metropolis algorithm calculates the new energy of the system. If the energy has decreased then the systems move to this state. If the energy has increased then the new state is established using the probability returned by the above formula. A certain number of iterations are carried out at each temperature and then the temperature is decreased. This is repeated until the system freezes into a steady state.

The following equation is directly used in simulated annealing, although it is usual to drop the Boltzmann constant as this was only introduced into the equation to cope with different materials. Therefore, the probability of accepting a worse state is given by the equation (Kirkpatrick et al 1983):

$P = \exp(-c/t) > r$

Where

c = the change in the evaluation function

t = the current temperature

r = a random number between 0 and 1

The probability of accepting a worse move is a function of both the temperature of the system and of the change in the cost function. It can be valued that as the temperature of the system decreases the probability of accepting a worse move is decreased. This is the same as slowly moving to a frozen state in physical annealing. Also note, that if the temperature is zero then only better moves will be accepted which effectively makes simulated annealing act like hill climbing. The following algorithm is mentioned by Russell, 1995.

Function SIMULATED-ANNEALING(Problem, Schedule) **returns** a solution state
Inputs :Problem, a problem
Schedule, a mapping from time to temperature
Local Variables :Current, a node
Next, a node
T, a "temperature" controlling the probability of downward steps

Current = MAKE-NODE(INITIAL-STATE[Problem])
For t = 1 **to** ∞ **do**
T = Schedule[t]
If Termination Condition **then**
return Current
Next = a randomly selected successor of Current
ΛE = fittness[Next] – fittness[Current]
if ΛE > 0 **then**
Current = Next

else if (exp(-ΛE/T) > probability **then**
Current = Next

2.2 Genetic Algorithm (GA)

Genetic algorithms are adaptive methods, which may be used to solve search and optimization problems (Beasley et al. (1993)). They are based on the genetic process of biological organisms. Over many generations, natural populations evolve according to the principles of natural selection, i.e. Survival of the fittest, first clearly stated by Charles Darwin in The Origin of Species. By simulating this process, genetic algorithms are able to evolve solutions to real world problems.

Before a genetic algorithm can be run, an appropriate encoding (or representation) for the problem must be developed. A fitness function is also required, which assigns a figure of merit to each encoded solution. During the run, parents must be selected for reproduction, and recombined to generate offspring. It is assumed that a solution to a problem may be represented as a set of parameters. These parameters (known as genes) are coupled together to form a string of values (chromosome). In genetic terminology, the set of parameters represented by a particular chromosome is referred to as an individual. The fitness of an individual depends on its chromosome and is evaluated by the fitness function.

The individuals, during the reproductive phase, are selected from the population and recombined, producing offspring, which involve the next generation. Parents are randomly selected from the population using a scheme, which favors fitter individuals. Having selected two parents, their chromosomes are recombined, typically using mechanisms of crossover and mutation. Mutation is usually applied to some individuals, to guarantee population diversity. The basic genetic pseudo code is as follows.

Genetic Algorithm
{
Generate initial population Pt
Evaluate population Pt
While stopping criteria not satisfied **Repeat**
{
Select elements from P_t to copy into P_{t+1}
Crossover elements of P_t and put into P_{t+1}
Mutation elements of P_t and put into P_{t+1}
Evaluate new population P_{t+1}
$P_t = P_{t+1}$
}

}

2.3 Proposed Hybrid Evolutionary Optimization Model

The following steps explains the Proposed SA-GA-Hybrid Model for solving JSSP

1. Initialize the solution X of size [p x (rxc)] (where r x c is the size of the JSSP and p is the population size)
2. For Each Generation Repeat 3 to 6
3. $X_{min} \leftarrow X_1$; $F_{min} \leftarrow$ Fittness(X_1) ; Current $\leftarrow X_1$
4. Do GA based perturbation on X and produce X_{new}
   T = Schedule[t]
   If SA_Termination_Condition **then**
   Go to 8
   X_{new}= GeneticOperationsOn(X)
5. Calculate Fitness of X_{new}
   F_{new}= Fittness(X_{new})
   Where
   F_{new} be the p x 1 matrix which will contain the individual fitness values.
6. Main SA based Check
   [F_{Sorted}, Index] $\leftarrow$ sort(F_{new}) ;
   Next $\leftarrow$ F_{Sorted} (top)
   ΛE = fittness[Next] – fittness[Current]
   if ΛE > 0 **then**
   Current$\leftarrow$ Next
   else if (exp(-ΛE/T) > probability **then**

Current ← Next

7. X(Index (top)) ← Current
   go to step 4
8. Finally return the results
   Makespan ← fittness[Current]
   BestSolution ← Current
   Stop.

The following is an example for a 3 x 3 JSSP.

Table 1 : A 3x3 JSSP

Job	Operations routing (processing time)		
1	1(3)	2(3)	3(3)
2	1(2)	3(3)	2(4)
3	2(3)	1(2)	3(1)

The one of the optimum schedule of the above problem is [$J_{1,1}$, $J_{3,1}$, $J_{2,1}$, $J_{1,2}$, $J_{3,2}$, $J_{2,2}$, $J_{2,3}$, $J_{1,3}$, $J_{3,3}$]. Here, for example, $J_{1,2}$ represents the operation 2 of the Job 1. Figure1shows one of such a optimum solution for the problem represented by "Gantt-Chart".

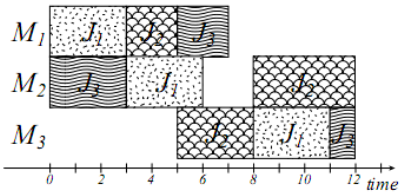

Figure 1 : The Gantt-Chart Representation of the Solution of the above 3x3 Problem

If we denote the operations of the job as follows,

Job1: Op1, Op2, Op3
Job2: Op4, Op5, Op6
Job3: Op7, Op8, Op9

Or simply

1 2 3
4 5 6
7 8 9

then, the schedule [$J_{1,1}$, $J_{1,2}$, $J_{1,3}$, $J_{2,1,}$ $J_{2,2}$, $J_{2,3}$, $J_{3,1}$, $J_{3,2}$, $J_{3,3}$] or simply [1, 2, 3, 4, 5, 6, 7, 8, 9] will be the one of the known worst case schedule which will satisfy all the conditions of the JSSP. But in this case, the makespan will not be optimum. The schedule [$J_{1,1}$, $J_{3,1}$, $J_{2,1}$, $J_{1,2}$, $J_{3,2}$, $J_{2,2}$, $J_{2,3}$, $J_{1,3}$, $J_{3,3}$] or simply [1, 7, 4, 2, 8, 5, 6, 3, 9] will be the one of the best know optimum solution. Here the strings "1, 2, 3, 4, 5, 6, 7, 8, 9" and "1, 7, 4, 2, 8, 5, 6, 3, 9" represents solutions and known as valid strings.

In GA, a legal string or a illegal string (of numbers) which represent the order of the schedule can be represented by a chromosome. For example, the known worst case solution can be represented as a chromosome of GA by a string "1, 2, 3, 4, 5, 6, 7, 8, 9". Similarly, the chromosome of GA "1, 7, 4, 2, 8, 5, 6, 3, 9" will represent a legal string which is an optimal solution of JSSP. And for example, the chromosome "3, 9, 4, 2, 1, 5, 6, 7, 8" will be a invalid string which correspond to a illegal operation or schedule since this schedule will not satisfy the conditions of JSSP.

So, if the initial selection of chromosomes of GA with random values, then there will be lot of invalid strings in the initial guess. The scope of the evolutionary algorithm is to permute the most optimal string to better most optimal string which will hopefully make that string as a legal string in proceeding generations/steps and finally it will end up with a string belongs to a better solution or optimal schedule with minimum makespan.

This assumption will be good and can produce meaningful solutions for lower order scheduling problems such as 3x3 JSSP or 4x4 JSSP. But, it may produce illegal solutions even after very long runs in the case of higher order scheduling problems like 15x15 JSSP. Because, if we randomly chose initial population then there will be much chance for getting all illegal strings in the initial set which belongs to no nearby solution. So the fitness calculation methods will lead to meaningless fitness values and the selection method will also be incapable of selecting a better solution in each generation or step. So in each step of the evolutionary process there will not be any guaranty of getting progressive solution.

So we believe that the random selection of initial solution/seed in an evolutionary algorithm will not lead to a better result in higher order scheduling problems such as 15x15 JSSP. So in this work, we have evaluated the performance of two evolutionary optimization techniques SA and GA with different initial conditions.

III. Results And Analysis

In the experiments, we have given the known worst case solution as initial "seed" for the evolutionary process. We expect that, the system will be capable of producing at latest one meaningful legal string in every generation/step/iteration and hence there will be a much good probability of achieving a better solution in the succeeding generations or steps.

3.1 Analysis of Performance with Different Problem Size

The GA was run for 100 generations with population size of 100. The SA was run for 3000 iterations since this simple SA will only handle one solution at a time.

The following figure shows a result (makespan=68) found by the GA based method for a 6x6 problem.

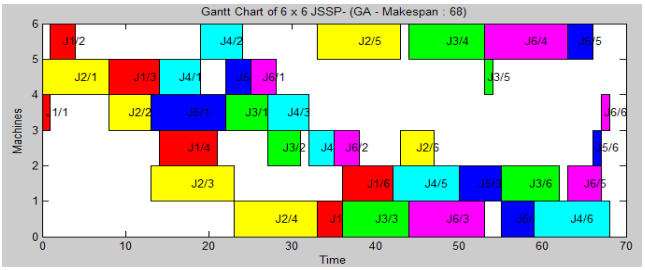

Figure 2 : Solution found by GA for a 6x6 Problem.

The following figure show a s result (makespan=77) found by the SA based method for a 6x6 problem.

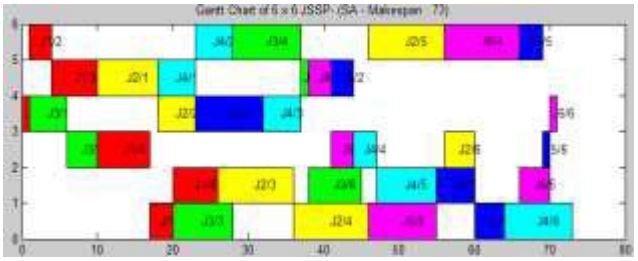

Figure 3 : Solution found by SA for a 6x6 Problem

The following figure shows a result (makespan=62) found by the SA-GA-Hybrid based method for the same 6x6 problem.

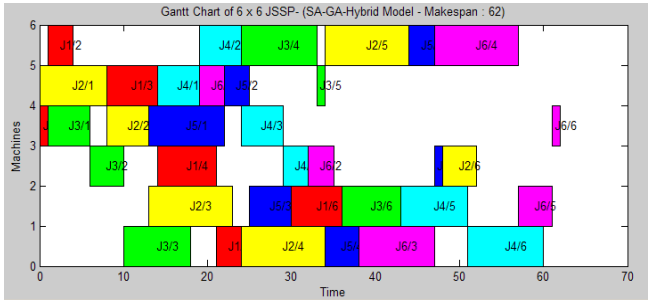

Figure 4 : A Solutions found by SA-GA-Hybrid for a 6x6 Problem .

The following figure shows another result (makespan=61) found by the SA-GA-Hybrid based method for the same 6x6 problem.

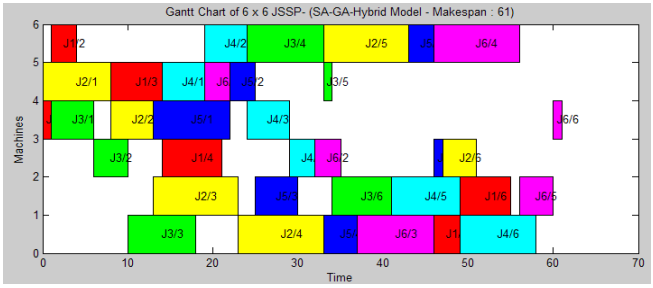

Figure 5: Another Solutions found by SA-GA-Hybrid for a 6x6 Problem

The Following fugures show the average fitness of the two methods over generations. If we see these plots, we can realize that both the algorithms behaves in a similar fashion.

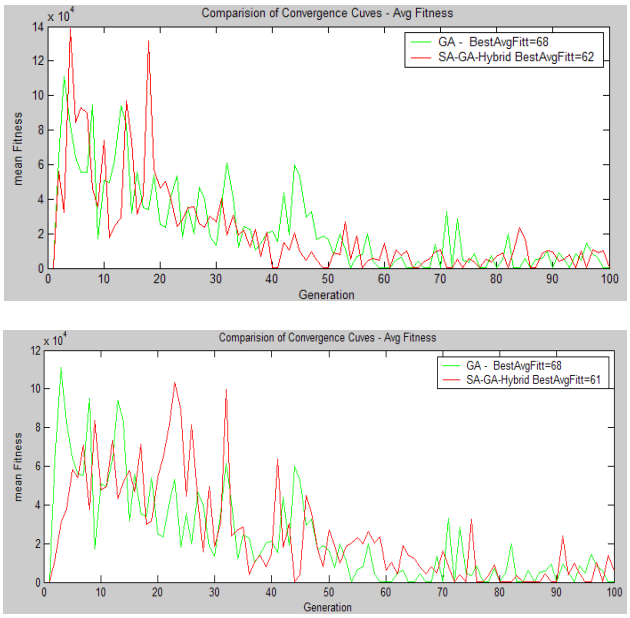

Figure 6 : The Average Fitness Plot of the Solutions found by GA and SA-GA-Hybrid algorithm for a 6x6 Problem .

The Following fugures show the best fitness of the two methods over generations. If we see these plots, we can realize that proposed SA-GA-Hybrid algorithm performes little bit better than the normal GA .

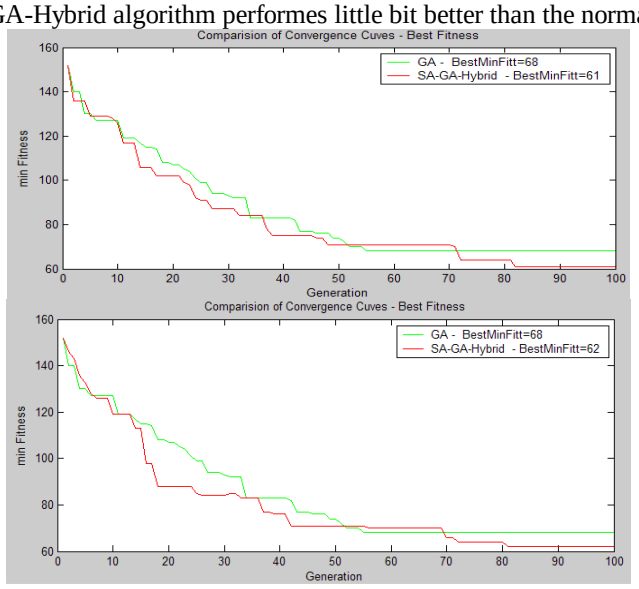

Figure 7 : The Best Fitness Plot of the Solutions found by GA and SA-GA-Hybrid algorithm for a 6x6 Problem

3.2 Performance in terms of speed

To measure the performance in terms of speed, the model was run with problems of different sizes and the results were tabulated as follows:

Table 2 : The time taken for different JSSP size

Sl.No	JSSP Size	Time Taken(sec)		
		GA	SA	SA-GA Hybrid
1	3x3	2.87	8.44	2.86
2	4x4	3.61	9.14	3.47
3	6x6	5.70	10.51	5.61
4	10x10	16.75	12.72	16.85
5	15x15	58.28	19.82	55.73

The following figure shows the performance in terms of time with respect to different JSSP problem size. If we carefully see the lines of GA and SA-GA Hybrid, then we can say that the performance of SA-GA Hybrid is little bit better than GA in some cases and almost equal in some cases.

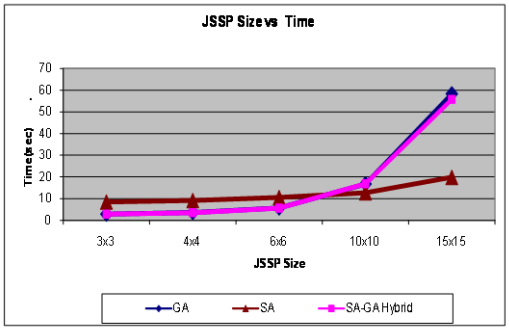

Figure 8 : JSSP Size vs Time Chart

3.3 Convergence Capability of the Algorithms

The convergence was measured in terms of makespan for different size of the problems was tabulated below. We only considered the convergence up to the first 100 iterations in the case of GA and SA-GA-Hybrid model during its run.

To measure the performance in terms of convergence, the model was tested with problems of different sizes and the results were tabulated as follows:

Table 3 : Startup with known worst case solution

Sl. No	JSSP		Achieved Optimal Solution (Makespan)		
	Size	Known optimum value	GA	SA	SA-GA Hybrid
1	3x3	12	12	12	12
2	4x4	272	272	286	272
3	6x6	55	68	72	61
4	10x10	902	2214	2899	2011
5	15x15	1268	6771	8241	6637

The following figure shows the performance in terms of Makespan with respect to different JSSP problem size.

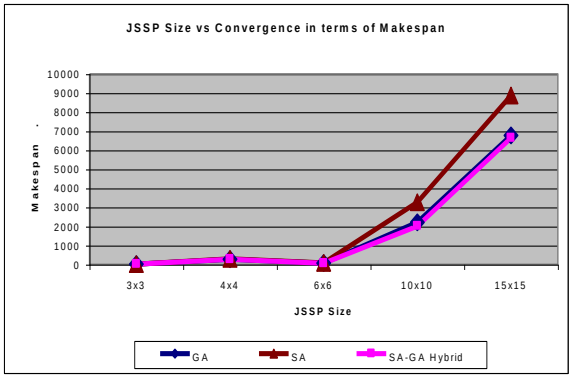

Figure 9: JSSP Size Vs Makespan Chart.

Even though they arrived solution is far from the optimal solution, the better performance in the case of SA-GA-Hybrid is very obvious.

IV. Conclusion And Future Work

The algorithms have successfully implemented two basic evolutionary models for solving JSSP using GA & SA, and also the proposed SA-GA-Hybrid. They arrived results showing that the models produced optimal or near-optimal solutions medium level job shop scheduling problems in a shot duration. While initializing with known, worst case solution, the evolutionary process was capable of converging into meaningful and more optimum solutions. Further, as shown in the convergence curves in the previous section, The SA-GA-Hybrid performed in a very better way than GA. Future works may address the issued involved in improving the performance of the SA-GA-Hybrid for high dimensional problems.

Acknowledgement

My sincere thanks to the management and principal of P.K.R. Arts College for Women, Gobichettipalayam and friends, who have motivated & helped me towards the topic and development of this research paper.

References

[1]. Aarts, E & Korst, J 1989, Simulated annealing and Boltzmann Machines, J.Wiley and Sons, Chichester.
[2]. Bandyopadhyay, S & Saha, S 2013, 'Some Single- and Multiobjective Optimization Techniques', Unsupervised Classification, Springer-Verlag Berlin Heidelberg.
[3]. Eswaramurthy, VP & Tamilarasi, A 2009, 'Hybridization of Ant Colony OptimizationStrategies in Tabu Search for Solving JobShop Scheduling Problems', International Journal of Information and Management Sciences, vol. 20, pp. 173-189.
[4]. Evolutionary computing edited by Ben Paechter, Napier University, <http://www.metaheuristics.net>
[5]. Moraglio , H.M.M. Ten Eikelder, R. Tadei, "Genetic Local Search for Job Shop Scheduling Problem", Technical Report, CSM-435 ISSN 1744-8050
[6]. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys. - Sequencing and scheduling: Algorithms and complexity. - In: S.C. Graves, A.H.G. Rinnoy Kan and P. Zipkin, editors, Handbooks in Operations Research and Management Science 4, North-Holland, 1993.
[7]. Dr. Daniel Tauritz, The abstract of the talk "Grand Challenges in Evolutionary Computing - Part II", Missouri S&T
[8]. Hongbo Liu, Ajith Abraham,Zuwen Wang, "A Multi-swarm Approach to Multi-objective Flexible Job-shop Scheduling Problems", School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China, Fundamenta Informaticae,IOS Press, 2009
[9]. José Fernando Gonçalves, Jorge José de Magalhães Mendes,Maurício G. C. Resende, "A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem", AT&T Labs Research Technical Report TD-5EAL6J, September 2002.
[10]. Kirkpatrick, S, Gelatt, C & Vecchi, M 1983, 'Optimization by simulated annealing,' Science, vol. 220, no. 4598, pp. 671-680.
[11]. Mahanim Binti Omar, "A Modified Multi-Step Crossover Fusion (Msxf) In Solving Some Deterministic Job Shop Scheduling Problem (Jssp), A thesis work submitted to Universiti Sains Malaysia, 2008
[12]. Y. Shi, R. C. Eberhart, Parameter selection in particle swarm optimization, in Evolutionary Programming VII: Proc. EP98, pp. 591-600 (New York: Springer-Verlag, 1998).
[13]. Takeshi Yamada and Ryohei Nakano, "Genetic Algorithms for Job-Shop Scheduling Problems", NTT Communication Science Labs, JAPAN, Proceedings of Modern Heuristic for Decision Support, pp.67, UNICOM seminar, March 1997, London.
[14]. Jayasankari**,** S & Tamilarasi, A 2012, 'Analysis of Two Stochastic Optimization Techniques for Solving Job Shop scheduling Problem', European Journal of Scientific Research, ISSN 1450-202X / 1450-216X, vol. 88, no. 3, pp.365-379, October.
[15]. Jayasankari, S, Tamilarasi, A & Thiagarasu, V 2013, 'Evaluation on PSObased Model for Solving JSSP', International Journal of Applied Research &Studies, vol. II, issue 2, February.
[16]. Jayasankari, S, Tamilarasi, A & Nasira, GM 2013, 'Evaluation on Simulated Annealing based Model for Solving Job Shop Scheduling Problem', IOSR Journal of Computer Engineering (IOSR-JCE) - e-ISSN: 2278-0661, p- ISSN: 2278-8727, vol. 9, issue 1, pp. 73-80, January - February.