Applied Soft Computing 49 (2016) 510–523

Contents lists available at ScienceDirect

Applied Soft Computing

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

Fast simulated annealing hybridized with quenching for solving job shop scheduling problem

Kashif Akram (Post Graduate Student)*, Khurram Kamal (Assistant Professor), Alam Zeb

Department of Mechatronics Engineering, College of Electrical and Mechanical Engineering, National University of Science and Technology (NUST), 46000, H12, Islamabad, Pakistan

ARTICLE INFO

Article history:
Received 20 January 2016
Received in revised form 19 July 2016
Accepted 21 August 2016
Available online 24 August 2016

Keywords:
Job-shop scheduling problem (JSSP)
Cauchy probability density function
Fast simulated annealing (FSA)
Hybridization
Makesspan
Quenching
Simulated annealing

ABSTRACT

Various heuristic based methods are available in literature for optimally solving job shop scheduling problems (JSSP). In this research work a novel approach is proposed which hybridizes fast simulated annealing (FSA) with quenching. The proposed algorithm uses FSA for global search and quenching for localized search in neighborhood of current solution, while tabu list is used to restrict search from revisiting previously explored solutions. FSA is started with a relatively higher temperature and as search progresses temperature is gradually reduced to a value close to zero. The overall best solution (BS) is maintained throughout execution of the algorithm. If no improvement is observed in BS for certain number of iterations then quenching cycle is invoked. During quenching cycle current temperature is reduced to nearly freezing point and iterations are increased by many folds, as a result of this change search becomes nearly greedy. The strength of the proposed algorithm is that even in quenching mode escape from local optima is possible due to use of Cauchy probability distribution and non-zero temperature. At the completion of quenching cycle previous values of search parameters are restored and FSA takes over, which moves search into another region of solution space. Effectiveness of proposed algorithm is established by solving 88 well known benchmark problems taken form published work. The proposed algorithm was able to solve 45 problems optimally to their respective best known values in reasonable time. The proposed algorithm has been compared with 18 other published works. The experimental results show that the proposed algorithm is efficient in finding solution to JSSP.

1. Introduction

The job shop scheduling problem (JSSP) is a well-known NP hard combinatorial optimization problem, as it is NP hard there is no guarantee of a global optimal solution in polynomial time. Job shops are very common in manufacturing industry and solving this problem optimally and in reasonable time have attracted many researchers. An optimal solution to JSSP optimizes machine utilization, reduces lead times, improves efficiency of manufacturing system and reduces costs. Due to complexity and broader application of solving JSSP, a large numbers of algorithms and approaches have been proposed by various researchers for many decades.

There are two basic approaches to solve JSSP; one is exact methods and the other is approximate methods. Exact methods are focused on finding precise solution to the problem. At successful completion, exact methods yield global optimal solution to the problem. Enumeration [1], integer programming [2], Lagrangian relaxation [3], dynamic programming [4], and branch and bound methods are a few examples of exact methods. Most effective and successful exact method is branch and bound (B&B) [1,5]. The basic principle of the B&B algorithm is to enumerate iteratively all feasible solutions one by one. For this purpose algorithm uses dynamically constructed tree structures [6]. Major drawback of B&B algorithm is that it tries to explore all possible solutions which take huge amount of time even for moderately smaller problems. All exact methods cannot solve larger problems in reasonable time. Therefore focus of research shifted to approximation based methods [7].

Approximation methods, as evident from their name, provide no guarantee of global optimal solution, but they can provide near optimal solution for moderate to complex problems, in moderate computing time.Some common approximation methods are priority dispatching, bottle neck heuristic, artificial intelligence, local search and metaheuristic methods.

* Corresponding author.
E-mail addresses: kashifakramnust@yahoo.com (K. Akram), k.kamal@ceme.nust.edu.pk (K. Kamal), alamzeb100@yahoo.com (A. Zeb).

http://dx.doi.org/10.1016/j.asoc.2016.08.037
1568-4946/

In priority dispatching methods a list of operations is formed based on some predefined priority rules. And operations are assigned to available machines or resources based on their position in priority list. These methods greatly reduce computation and processing times [8,9]. But these methods do not focus on reducing global make span. Therefore, efficacy of these methods greatly reduces for larger problems [10].

Shifting bottleneck procedure (SBP) is an approximation method based on bottleneck iterative heuristics [11,12]. SBP is an iterative method, it adds schedule of one additional machine in subsequent iterations, and re-optimization is used to adapt schedule for already scheduled machines. This method is difficult to implement and program [10].

Since 1980s, with advancements in computer hardware and software, new techniques are being developed for solving JSSP and artificial intelligence (AI) is one of them. AI based algorithms are inspired from nature and bio systems. These algorithms not only use quantitative knowledge but also employ qualitative information to solve problems. AI is able to generate and use heuristics which are more complex than dispatching rules. Artificial neural network (ANN) is one of the methods to solve JSSP [13]. ANN is a software based imitation of biological brain. Brain consists of a large number of neurons which are interconnected in a very complex way. Various researchers have explored application of ANN to solving JSSP [14–17]. Implementing AI algorithms is difficult and time consuming, and these techniques are often problem specific [10].

Local search methods [18–28] are another approximation based techniques to solve JSSP. Basic theme of these techniques is to modify existing solution by searching in neighborhood of this solution. A neighborhood operator is used to generate a new solution which is expected to be better than its seed.

Like all other algorithms and approaches, there are pros and cons of every optimization methods. Therefore combinations of two or more techniques are used to solve JSSP. In this way weaknesses of one method can be supplemented by the strengths of another. These methods are called hybrid algorithms. Hybrid methods are frequently employed for solving JSSP for example hybrid GA and TS [29–31], hybrid PSO with variable neighborhood search (VNS) [32], and hybrid Artificial Immune System (AIS) with PSO [10]. The hybrid algorithms perform better than its corresponding individual counterparts as with hybridization convergence rate is usually high and it also helps in escaping local minima. All the algorithms developed for JSSP have their strengths and weaknesses [10]. Therefore, researchers are constantly in search of new algorithms and a lot of efforts have been put in to optimize and improve existing methods.

It has been observed by various researchers that hybridization of simulated annealing (SA) is an efficient method to solve JSSP; For example, Xia and Wu [33] have merged PSO with SA to solve JSSP. Their algorithm uses local search property of particles by self-experience and global search property by neighboring experience to find the global optimal solution. SA is used by them as a means to avoid falling in local minima. Zhang and Wu [34] have hybridized simulated annealing with artificial immune system to solve JSSP using weighted tardiness as an objective function with the core idea of identification of bottleneck jobs. Tamilarasi and Kumar [35] have hybridized GA with SA; GA is used to search globally and if best solution does not improve over a period of time SA is used to escape local minima. Thamilselvan and Balasubramanie [36] have added another dimension by hybridizing three methods GA, TS and SA, and have produced good results for large size problems. They have integrated GA with TS in reproduction phase and SA is used to avoid early convergence to local minima. All the above researches have used SA, either as an improving tool for raw results or a way to avoid local optimal solutions. Akram and Kamal [37] have proposed an algorithm which used SA as a primary search tool and hybridized it with quenching (a material treatment process where temperature is suddenly reduced to impart certain characteristics). Their algorithm invokes a quenching cycle when no improvement in best solution is made by SA over a predefined number of iterations. During this quenching cycle temperature (SA) is suddenly reduced and iterations are increased by many folds. After completion of quenching cycle temperature and numbers of iterations are restored to their previous values, generating promising results. In this paper hybrid algorithm of Akram and Kamal [37] has been improved. They [37] have used Boltzmann probability density function (PDF) to accept or reject a poor (non-improving) solution and a geometric cooling schedule to reduce temperature (SA) whereas in the proposed research work, a novel technique (HFSAQ) is presented which hybridize fast simulated annealing (FSA) with quenching. In HFSAQ, Cauchy PDF is used in place of Boltzmann PDF and a faster inversely linear temperature reduction schedule is used in place of geometric reduction of temperature. In this way capability of Akram and Kamal [37] algorithm has been enhanced. This approach has been tested on a number of benchmark problems, taken from literature [39], for solution accuracy and computational efficiency.

The rest of the paper is organized as follows; Section 2 covers conventional problem definition. Section 3 provides details of the proposed hybrid method. Section 4 discusses implementation of the algorithm and its results on benchmark problem followed by conclusion in the last section.

2. Problem definition

In traditional terms the job shop scheduling problem (JSSP) can be defined as a system on n jobs which are to be processed on m machines, where each job have m operations and each operation has predefined fixed order of execution.The aim of any scheduling algorithm is to schedule each and every job on respective machines while considering some scheduling objective. In our study we have chosen minimization of make span as a scheduling objective. To simplify, following assumptions are made: (1) all the machines are free and all the jobs can be loaded on machines at the beginning of the execution of the schedule (2) One machine can only process one job at time and one job cannot be processed simultaneously on more than one machine. (3) Once a job is loaded on a machine, it will not be interrupted until it is completely processed. (4) Processing times and precedence constraints are static and predefined. (5) There is no constraint on queuing and storage time. (6) Factors such as setup times, transportation, costs, release times, due dates, machine breakdown and operator unavailability etc are ignored. Any system fulfilling above conditions is also called a static JSSP.

In mathematical terms, for JSSP, each job consists of m operations $O_{j1}, O_{j2},\ldots,O_{jm}$. Required machine and processing time for each operation O_{jk} is represented by Π_{jk} and T_{jk} respectively. Π and T are two matrices with dimensions nxm. Hence two values n and m, and two matrices Π and T are able to fully describe a static JSSP. Final output of any scheduling algorithm is an nxm matrix S, where an element S_{jk} is the starting time of operation O_{jk}. JSSP can be formulated as;

$$\text{Min} \quad \text{Max}\,(S_{jk} + T_{jk} \;:\; 1 \le j \le n \text{ and } i \le k \le m) \tag{1}$$

Subjected to

$$S_{jk} + T_{jk} \le S_{j(k+1)} \;:\; 1 \le j \le n \text{ and } i \le k \le m-1 \tag{2}$$

$$S_{jk'} + T_{jk'} \le S_{j'k} \text{ or } S_{j'k} + T_{j'k} \le S_{jk'} :$$

$$\Pi_{jk'} = \Pi_{j'k} \text{ and } 1 \le j \text{ and } j' \le n \text{ and } 1 \le k \text{ and } k' \le m \tag{3}$$

First constraint ensures precedence requirements and second constraint avoids simultaneous loading of jobs on machines. The

Scheduling Data

	Process Time			Machine Number		
Job	O1	O2	O3	O1	O2	O3
1	7	6	6	1	2	3
2	6	7	7	2	3	1
3	8	5	9	3	2	1

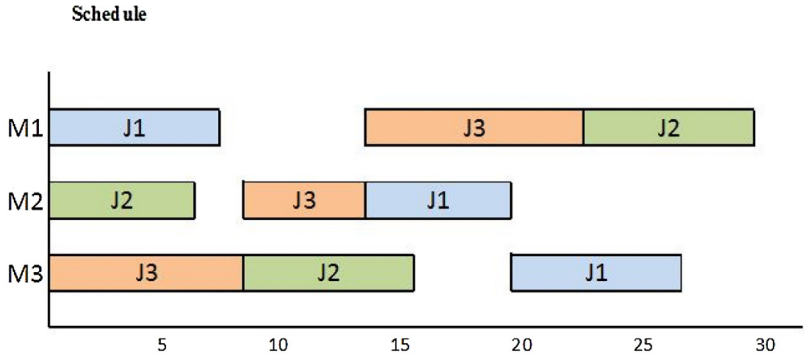

Fig. 1. A typical job shop and its schedule.

above formulation is called disjunctive graph formulation [11]. A sample JSSP problem and its solution are shown in Fig. 1.

3. Proposed hybrid method

This paper purposes a novel hybrid approach (HFSAQ) which combines fast simulated annealing (FSA) with quenching. FSA is a variant of simulated annealing which uses different cooling approach and probability density function (PDF). In following sections technological details such as solution representation, neighborhood generation strategy, simulated annealing, fast simulated annealing and proposed hybrid algorithms are discussed.

3.1. Solution representation

In this research work permutation with repetition proposed by Bierwirth [40]) is used for solution representation.In this representation every solution is represented by a one dimensional array of numbers. The length of array is equal to the product of number of jobs and number of machines (operations). For example if there are 3 jobs and 3 machines (operations) then length of solution array will be 9. Each location will have a number equal to or greater than 1 and less than or equal to total number of jobs. This array represents order of execution of operations in job shop. A detailed description of solution is shown in Fig. 2.

The key advantage of permutation with repetition is that every solution is a feasible solution and no repair strategy is required. On the down side many different permutations results in same schedule, but its coding and decoding efficiency makes it a desirable solution representation option for researchers. The solution permutation is decoded into a semi-active schedule. In this type of schedule, without changing processing order, no operation can be finished earlier. Decoding algorithm is designed in such a way that each machine is assigned, in the beginning, an availability time slot, this time slot is used to assign start times to the operation which are to be scheduled. Default duration of the time slots is equal to the sum of the processing times of all the jobs in JSSP. As the scheduling process goes on these time slots are sub divided and some of the sections are booked for operations and deleted from the slots, in this way it is ensured that multiple jobs could not be loaded simultaneously. The algorithm also keeps track of earliest possible starts time for each job; this data is updated after scheduling of each operation.

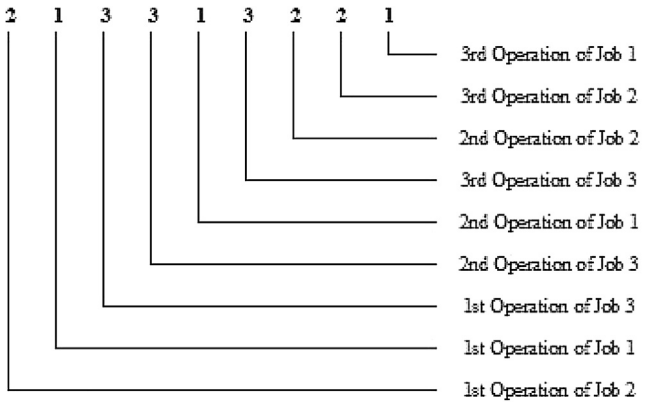

Fig. 2. Solution Representation.

Keeping track of earliest possible start time of each job is vital to ensure precedence constraint. When decoding algorithm finishes its run it provides both makespan and schedule at the same time. An illustrated example is shown in Fig. 3a is the JSSP data which is to scheduled, sum of its all the operation is 12, and therefore at the beginning all the machines are assigned an availability time slot of duration 12. Fig. 3b shows the solution permutation which is the order in which operations are to be scheduled. For example first number is 1 which tells first operation of job 1 is to be scheduled first and then so on. 3c shows the step by step progress of decoding process and status of time slots, earliest start times and schedule.

3.2. Neighbourhood generation

FSA is a neighborhood search algorithm; therefore a neighborhood operator is required to generate feasible neighbors. Neighborhood generation methodology proposed by Nowicki and Smutnicki [41] has been used for the proposed algorithm. Makespan of existing schedule can only be reduced by swapping operations which are on critical path; therefore critical path is estimated by the algorithm proposed by Cruz-chavez and Frausto-Solis [42]. Then this critical path is decomposed into blocks; each block contains successive operations which are to be performed on same machine. Nowicki and Smutnicki proposed that interchange should be performed near the borderline of blocks, on any ran-

JSSP Data to be scheduled

	Operation 1		Operation 2	
Job	M/C	Time	M/C	Time
1	1	3	2	1
2	2	2	1	2
3	1	1	2	3

a

Solution permutation to be schedule

1	1	2	2	3	3

b

Step by step description of execution of algorithm to decode chromosome into a semi active schedule

	Operation to be scheduled				
Index	Machine	Time	Time Slots after assigning start time to operation	Job's Earliest Start Times	Schedule*
	At Beginning		M1» (0,12] M2» (0,12]	J1=0 J2=0 J3=0	X X X X X X
1	1	3	M1» (3,12] M2» (0,12]	J1=3 J2=0 J3=0	0 X X X X X
2	2	1	M1» (3,12] M2» (0,3], (4,12]	J1=4 J2=0 J3=0	0 3 X X X X
3	2	2	M1» (3,12] M2» (2,3], (4,12]	J1=4 J2=2 J3=0	0 3 0 X X X
4	1	2	M1» (5,12] M2» (2,3], (4,12]	J1=4 J2=5 J3=0	0 3 0 3 X X
5	1	1	M1» (6,12] M2» (2,3], (4,12]	J1=4 J2=5 J3=6	0 3 0 3 5 X
6	2	3	M1» (6,12] M2» (2,3], (4,6], (9,12]	J1=4 J2=5 **J3=9**	0 3 0 3 5 6

* X represents a vacant location where a value will be assigned in subsequent iterations
J3=9 is also the makespan of the schedule

c

Fig. 3. A solved illustrated example of decoding JSSP a) JSSP data b) Solution permutation c) step by step execution and status of key variables.

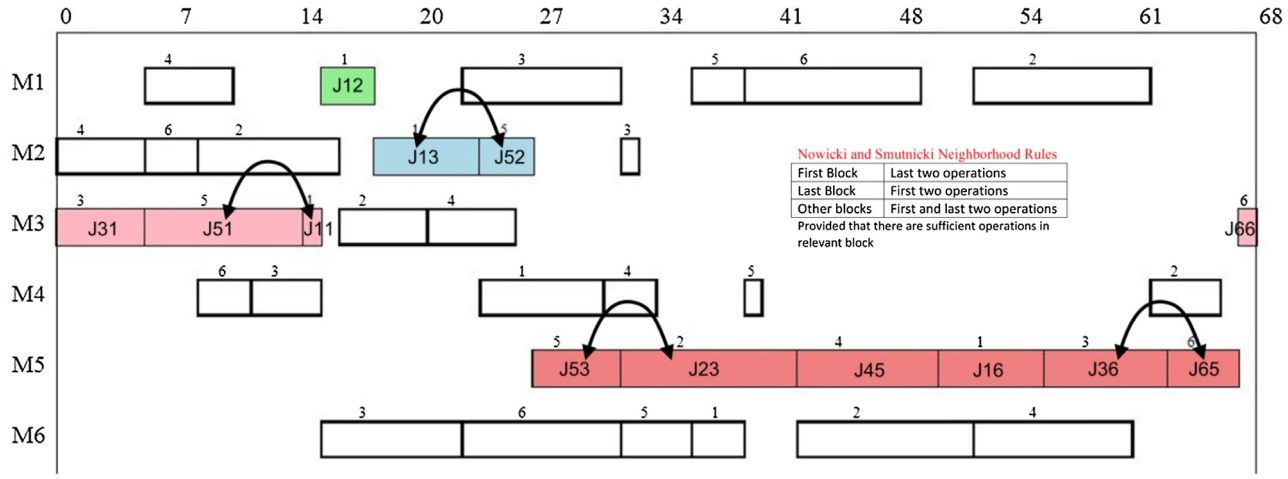

Fig. 4. Critical path-decomposed into blocks.

domly selected critical path. To generate neighborhood, from the first block last two operations are swapped, from last block first two operations are swapped, and from the rest first two and last two operations are swapped, provided that the block has enough operations. A significant property of Nowicki and Smutnicki neighborhood is that if all the operations on critical path are in one block or every block contains only one operation then the current solution is optimal. Nowicki and Smutnicki neighborhood is further explained by Fig. 4, a system of 6 jobs and 6 machines is scheduled and its Gantt chart is shown, the operations on critical path are highlighted (in color). From the figure; it can be observed that this critical path can be decomposed into 5 blocks. As per Nowicki

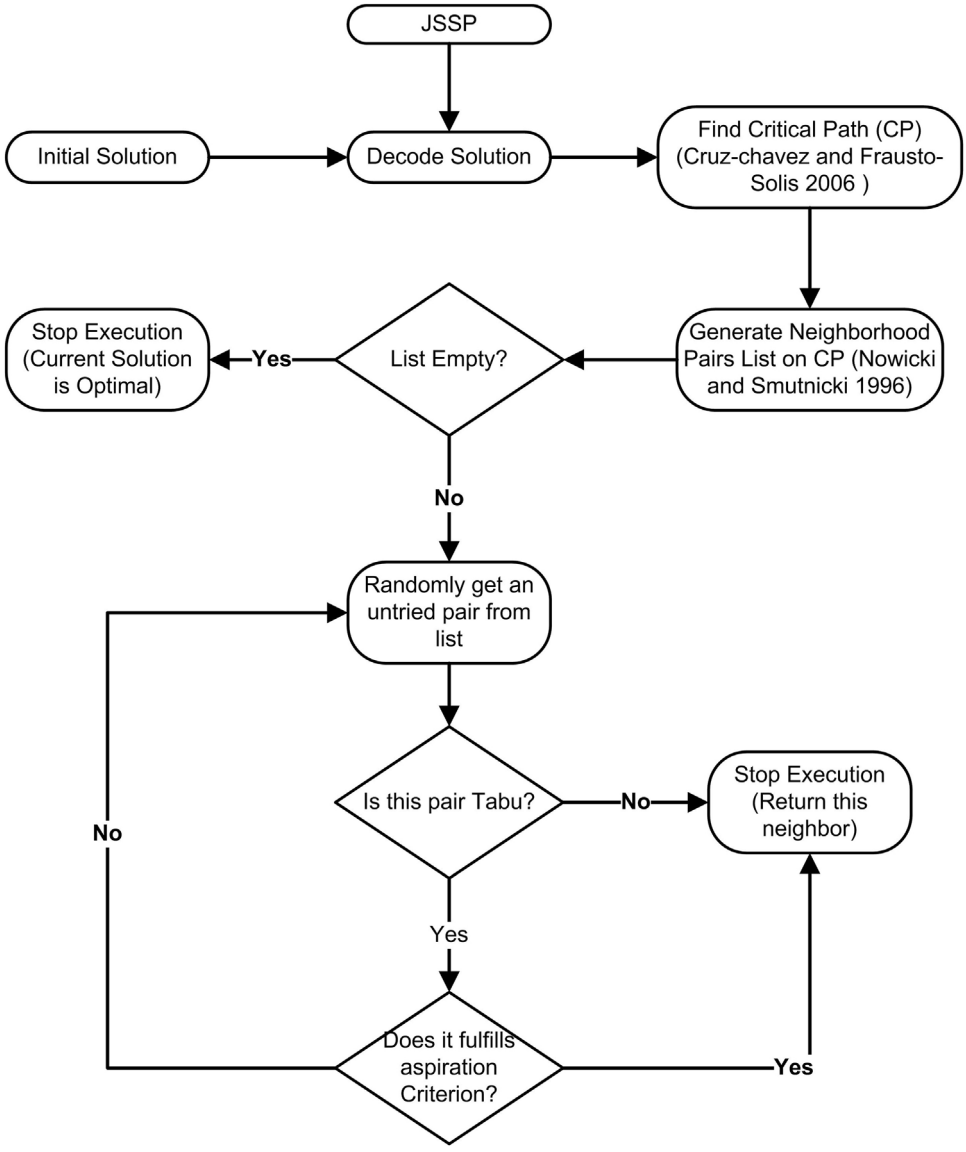

Fig. 5. Neighbourhood Generation.

and Smutnicki pairs J51-J11, J13-J52, J53-J23 and J36-J65 make up the neighborhood.

The detailed process of neighborhood generation is explained in flowchart shown in Fig. 5. Candidate pairs of neighborhood swaps are produced by Nowicki and Smutnicki, then from these a pair is randomly selected which is not tabu and returned to main algorithm for further processing.

3.3. Simulated annealing (SA)

SA is an optimization algorithm inspired from a physical process of material treatment [43]. Optimization is started with a high temperature and then it is gradually reduced to a low value by the use of geometric cooling schedule. A starting solution is generated and set as current solution. To move from current solution to another, a neighboring solution is generated. If neighboring solution is better than the current solution then it is set as current solution otherwise Boltzmann PDF is used to accept a poor solution. The most significant feature of SA is its ability to allow uphill (bad) moves under certain probability. The acceptance probability of bad moves depends upon current temperature and difference in fitness values of current and neighboring solutions. As temperature decreases or difference in fitness is large, the acceptance probability of uphill moves decreases and vice versa, in this way SA escapes local optima. Flow chart of SA is shown in Fig. 6

3.4. Fast simulated annealing (FSA)

FSA is a variant of conventional simulated annealing and is a semi local search method with occasional long jumps [38]. It has used Cauchy's PDF which is a fat tail probability distribution. With a fat tail distribution there is relatively high probability of generating large change in output with small change in inputs. Therefore Cauchy's PDF is more sensitive to changes in fitness of current and neighboring solution than Boltzmann's PDF. Due to this property, FSA is less likely to be trapped in local optima. It uses inversely linear cooling schedule which is much faster than geometric cooling rate of SA. FSA explores more search space at faster rate than conventional SA and it is more adept in avoiding local minima. Flow chart of FSA is shown in Fig. 7, where blocks different than SA are highlighted

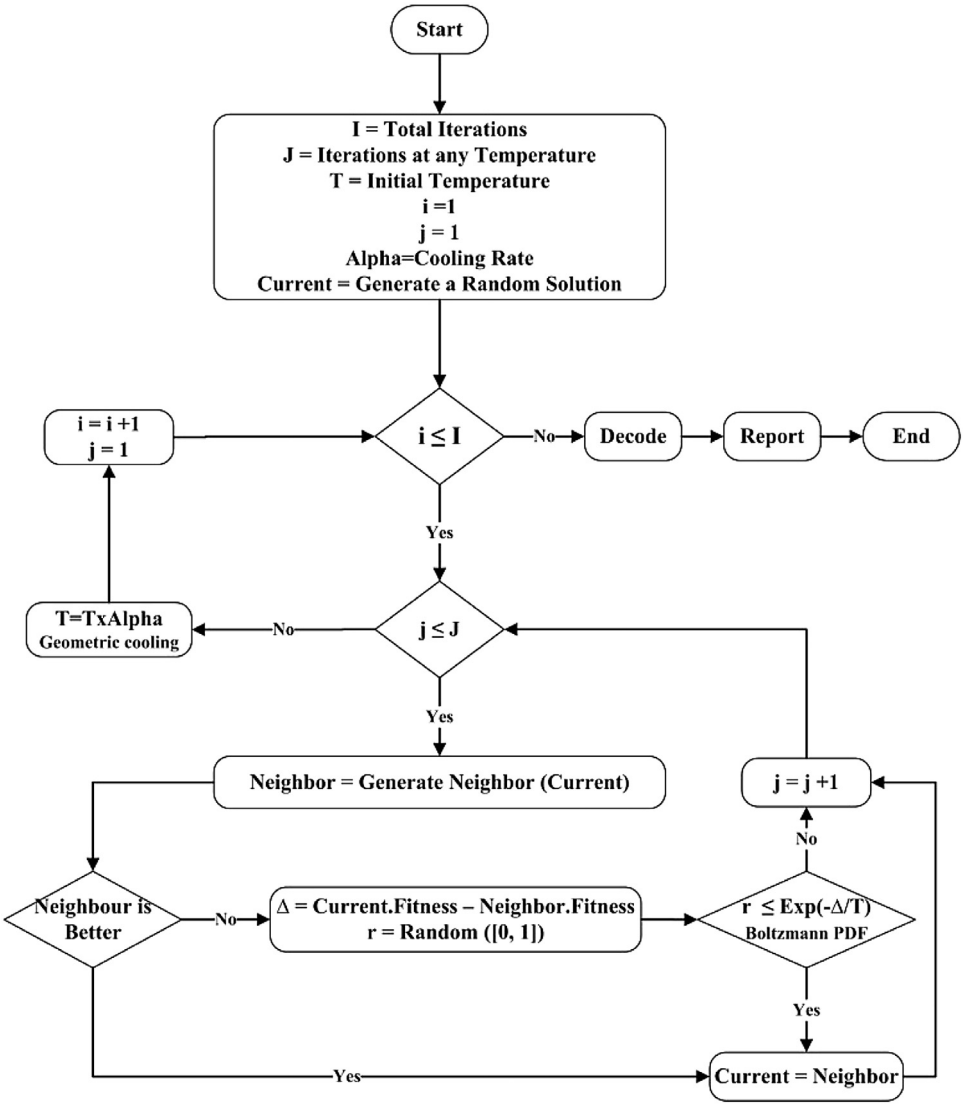

Fig. 6. The flow chart for SA.

3.5. Proposed method—hybridization of fast simulated annealing with quenching (HFSAQ)

In case of conventional SA or FSA, temperature is reduced form high value to a low value using cooling schedule. To achieve optimum results, at every temperature step, a fixed number of iterations are executed. In proposed novel approach, if fitness of the previously known best solution (BS) is not improved for a number of iterations (predefined) then a quenching cycle is introduced to perform a local search in solution hyper plane. Flow chart of the proposed hybrid method (HFSAQ) is shown in Fig. 8, where blocks different than FSA are highlighted.

A legal random sequence of operations (for details see Section 3.1) is generated and used as an initial solution, and the solution is further improved by proposed algorithm. Rapid annealing parameters are assigned their initial values. A list of neighborhood solutions (moves) is generated by using suitable neighborhood structure (see Section 3.2), then a solution is randomly selected and checked against tabu list, if the solution is found tabu then another solution is picked from the available list, otherwise it is taken as candidate neighbor solution and its fitness is calculated. If this solution is better than the current solution then it is accepted and current solution is replaced by it. Otherwise Cauchy probability distribution (density) function is invoked to accept or reject the poor solutions; this decision is based on the difference in feasibilities of current and neighborhood solutions and value of current temperature. An overall best solution (BS) is maintained. If this BS does not change for predefined number of iterations then a quenching cycle is brought into play which significantly reduces current temperature (value T see Fig. 6) and increases iterations at same temperature (value J see Fig. 6) by many times. At the end of this quenching cycle number of iterations at same temperature (J) is returned to its default value and temperature (T) is restored to its previous value, the cycle is repeated till stopping criteria is met. Stopping criteria for our proposed algorithm is either finding of lower bound of the problem or total number of iterations run out.

Incorporation of tabu list helps the algorithm by restricting the search to revisit recently explored solutions. The tabu list stores the attributes of moves rather than the attributes of the solutions, the main advantage of this setup is reduced computation time, but on the downside itmay declare an unexplored solution as tabu. To overcome this problem an aspiration criterion is used which accepts a tabu move provided that its makespan is less than the current best makespan found so far. The stored attributes are sequence of operations and their respective positions on machines [44]; in this way a legal solution is less likely to be declared as tabu. When-

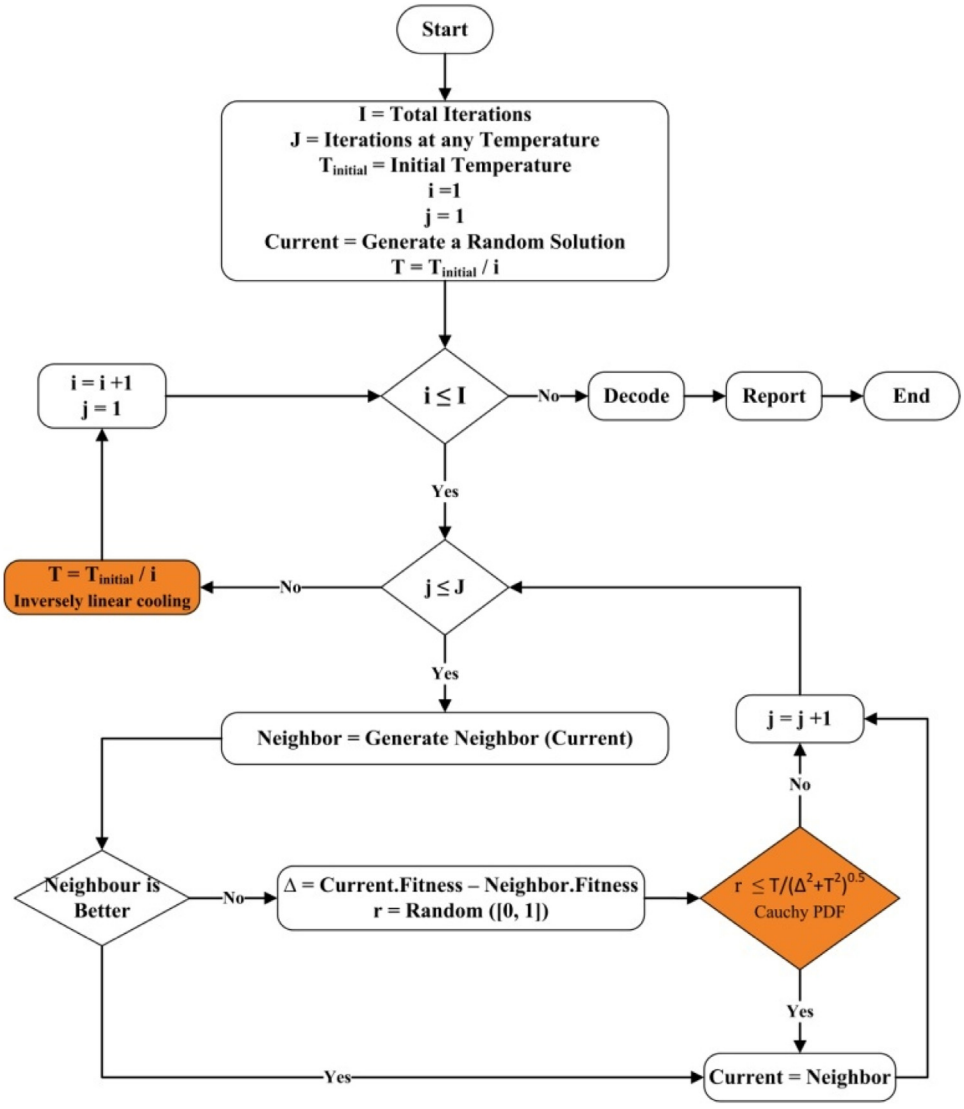

Fig. 7. The flow chart for FSA.

Table 1
Parameters for FSA and HFSAQ.

S/N	Parameter	FSA	During Quenching Cycle
1	Maximum Iteration	2000	NA
2	Maximum Iteration at one temperature	500	5000
3	Initial Temperature	0.5	Existing temp is reduced by 50 times

ever a neighboring solution is accepted as current solution, tabu list is reset to empty. If the length of tabu list is too short then cycling cannot be avoided and if length is too long then it put over restrictions on the search and many best solutions may be missed. For our proposed algorithm we have used circular tabu list with dynamic empirical length L = 10 + n/m; where n = number of jobs and m = number of machines.

In this method FSA is used for global search while intense local search is performed by quenching cycle. During this quenching cycle an intense local search is perform in neighborhood of current solution. Innovation of this quenching cycle is that it is nearly greedy (not absolutely greedy), which means even during local intense search some bad solutions, due to Cauchy's PDF, will be accepted which increases probability of getting out of local minima. When quenching cycle ends parameters are again relaxed and search moves to some other part of solution space, and then perform a new quenching cycle (local intense search) in some new location. The overall effect is that there is less likelihood of getting struck in local optima and more thorough search of solution space is performed.

4. Experimental results and discussions

Performance of proposed HFSAQ algorithm for solving JSSP is evaluated by testing problems taken from OR-library [39], selecting 88 benchmark problems from five classes of JSSP. Selected problems are FT06, FT10 and FT20 from Fisher and Thompson [45], LA01 to LA30 from Lawrence [46], ORB01 to ORB10 form Applegate and Cook [47], ABZ5 to ABZ9 from Adams et al. [11], and TA11 to TA50 from Taillard [48]. 27 out of 88 selected problems are square, which are considered more difficult than rectangular problems by research community. Algorithm is coded in visual C#.net as single threaded application and run on PC with Intel(R) Core(TM) 2 Quad CPU 2.4 GHz. Physical memory installed is 4GB. The proposed system was able to solve 45 problems optimally. The parameters used for FSA and HFSAQ are shown in Table 1, which were finalized after rigorous testing on benchmark problems.

Table 2
(a) Experimental results for solving different problem with FSA and HFSAQ. (b) Experimental results for solving TA11 to TA50 problem with FSA and HFSAQ.

S/N	Prob-lem	Fast simulated annealing					Proposed Method – HFSAQ					Time Ratio HFSAQ to FSA	Improv-ement(%)
		Worst	Average	Best	ARD	Average Time	Worst	Average	Best	ARD	AverageTime		
1	**ft06**	60	56	55	1.82	0.05	55	55	55	0.00	0.03	0.7	0.0
2	**ft10**	967	949.6	935	2.11	18.30	936	932.4	930	0.00	30.46	1.7	0.5
3	**ft20**	1180	1175.8	1173	0.93	20.04	1178	1167.6	1165	0.00	37.04	1.8	0.7
4	**la01**	666	666	666	0.00	0.08	666	666	666	0.00	0.09	1.1	0.0
5	**la02**	655	655	655	0.00	2.22	655	655	655	0.00	2.65	1.2	0.0
6	**la03**	597	597	597	0.00	3.17	597	597	597	0.00	4.47	1.4	0.0
7	**la04**	590	590	590	0.00	3.84	590	590	590	0.00	4.23	1.1	0.0
8	**la05**	593	593	593	0.00	0.00	593	593	593	0.00	0.00	1.3	0.0
9	**la06**	926	926	926	0.00	0.01	926	926	926	0.00	0.02	2.6	0.0
10	**la07**	890	890	890	0.00	0.28	890	890	890	0.00	0.20	0.7	0.0
11	**la08**	863	863	863	0.00	0.17	863	863	863	0.00	0.22	1.3	0.0
12	**la09**	951	951	951	0.00	0.04	951	951	951	0.00	0.05	1.3	0.0
13	**la10**	958	958	958	0.00	0.02	958	958	958	0.00	0.03	1.8	0.0
14	**la11**	1222	1222	1222	0.00	0.08	1222	1222	1222	0.00	0.62	7.9	0.0
15	**la12**	1039	1039	1039	0.00	0.06	1039	1039	1039	0.00	0.08	1.2	0.0
16	**la13**	1150	1150	1150	0.00	0.15	1150	1150	1150	0.00	0.10	0.7	0.0
17	**la14**	1292	1292	1292	0.00	0.00	1292	1292	1292	0.00	0.00	0.9	0.0
18	**la15**	1207	1207	1207	0.00	0.43	1207	1207	1207	0.00	0.49	1.1	0.0
19	**la16**	979	966.8	956	2.31	15.95	945	945	945	0.00	27.57	1.7	1.2
20	**la17**	784	784	784	0.00	14.77	784	784	784	0.00	26.60	1.8	0.0
21	**la18**	861	853	848	0.59	14.30	848	848	848	0.00	26.60	1.9	0.0
22	**la19**	849	846.2	842	0.50	11.20	842	842	842	0.00	22.28	2.0	0.0
23	**la20**	907	906	902	0.44	13.48	907	903	902	0.00	25.87	1.9	0.0
24	**la21**	1070	1062.6	1057	1.59	37.02	1048	1046.8	1046	0.00	51.21	1.4	1.0
25	**la22**	946	942.2	935	1.64	38.63	930	927.6	927	0.00	52.21	1.4	0.9
26	**la23**	1032	1032	1032	0.00	2.24	1032	1032	1032	0.00	2.63	1.2	0.0
27	**la24**	955	947.4	942	1.33	32.46	937	936.6	935	0.00	47.60	1.5	0.7
28	**la25**	993	990.6	988	1.39	36.15	984	978.4	977	0.00	51.59	1.4	1.1
29	**la26**	1218	1218	1218	0.00	20.68	1218	1218	1218	0.00	24.97	1.2	0.0
30	**la27**	1277	1269	1264	2.75	62.70	1240	1237	1235	0.00	83.91	1.3	2.3
31	**la28**	1234	1224.6	1218	0.71	60.72	1221	1217.4	1216	0.00	82.42	1.4	0.2
32	**la29**	1215	1199.4	1188	3.66	67.46	1170	1167.2	1164	0.88	86.11	1.3	2.0
33	**la30**	1355	1355	1355	0.00	8.43	1355	1355	1355	0.00	8.54	1.0	0.0
34	**orb01**	1101	1086	1064	2.55	20.78	1064	1060	1059	0.00	30.94	1.5	0.5
35	**orb02**	897	894.8	889	0.77	19.13	889	888.6	888	0.00	24.46	1.3	0.1
36	**orb03**	1078	1047.6	1022	4.24	23.25	1005	1005	1005	0.00	31.22	1.3	1.7
37	**orb04**	1035	1025.4	1018	2.03	24.81	1007	1005.4	1005	0.00	31.58	1.3	1.3
38	**orb05**	914	901	894	1.58	22.48	888	887.6	887	0.00	28.93	1.3	0.8
39	**orb06**	1031	1022.2	1013	1.21	20.92	1012	1010.8	1010	0.00	31.46	1.5	0.3
40	**orb07**	406	401.2	397	1.06	18.90	397	397	397	0.00	26.01	1.4	0.0
41	**orb08**	933	926.2	919	3.03	21.61	902	899.6	899	0.00	30.60	1.4	2.2
42	**orb09**	947	940.6	934	0.71	24.89	944	941.4	934	0.00	31.93	1.3	0.0
43	**orb10**	952	947.8	944	0.40	17.36	946	944.8	944	0.00	22.83	1.3	0.0
44	**abz5**	1249	1242.4	1239	0.68	5.22	1242	1237.4	1234	0.00	6.18	1.2	0.4
45	**abz6**	966	951.8	947	0.93	5.16	957	947.4	943	0.00	5.58	1.1	0.4
46	**abz7**	705	695.8	691	6.07	24.01	681	679.9	676	2.74	34.43	1.4	4.2
47	**abz8**	726	712.6	705	7.16	25.25	683	680.4	678	1.95	33.41	1.3	4.0
48	**abz9**	738	730.6	721	7.76	25.09	702	695	693	2.21	32.55	1.3	4.3
Average Relative Difference					1.29	Average Relative Difference				0.16	Average	1.5	0.64%

S/N	Prob-lem	Fast simulated annealing					Proposed Method – HFSAQ					Time Ratio HFSAQ to FSA	Improv-ement (%)
		Worst	Average	Best	ARD	AverageTime	Worst	Average	Best	ARD	Average Time		
1	**TA11**	1431	1423.9	1410	6.58	154.5	1377	1370.4	1361	2.87	178.9	1.16	3.48
2	**TA12**	1424	1412.6	1404	3.92	134.0	1382	1376.9	1372	1.55	155.1	1.16	2.28
3	**TA13**	1395	1384.3	1375	7.25	165.2	1376	1360.1	1342	4.68	176.2	1.07	2.4
4	**TA14**	1391	1377.2	1375	2.23	160.9	1351	1348.7	1345	0	192.2	1.19	2.18
5	**TA15**	1405	1396.9	1387	6.37	199.2	1353	1350.1	1340	2.76	219.9	1.1	3.39
6	**TA16**	1425	1407.4	1398	7.21	190.6	1375	1368.3	1360	4.29	208.2	1.09	2.72
7	**TA17**	1515	1505.1	1496	2.33	207.7	1479	1473.7	1464	0.14	222.7	1.07	2.14
8	**TA18**	1437	1427	1422	3.87	167.8	1416	1410.8	1400	2.26	178.3	1.06	1.55
9	**TA19**	1421	1409.7	1398	7.20	192.3	1351	1343.2	1339	2.68	214.5	1.12	4.22
10	**TA20**	1431	1422.5	1416	7.44	213.1	1360	1358	1353	2.66	228.5	1.07	4.45
11	**TA21**	1692	1679.8	1672	6.29	249.5	1654	1652.7	1646	4.64	283.5	1.14	1.56
12	**TA22**	1634	1621.4	1607	4.22	304.9	1610	1607.1	1606	4.15	363.3	1.19	0.06
13	**TA23**	1603	1592.9	1581	7.26	366.8	1567	1565.4	1564	6.11	451	1.23	1.08
14	**TA24**	1715	1699.1	1683	4.79	290.1	1667	1656.8	1648	2.62	340.6	1.17	2.08
15	**TA25**	1674	1653.8	1637	7.84	385.8	1617	1608.8	1599	5.34	419.6	1.09	2.32
16	**TA26**	1724	1700.2	1687	8.28	425.2	1673	1659.5	1647	5.71	481.6	1.13	2.37
17	**TA27**	1765	1754.2	1742	7.73	343.4	1701	1691.2	1684	4.14	401.3	1.17	3.33
18	**TA28**	1674	1660.6	1648	3.58	243.9	1625	1621.9	1613	1.38	296.8	1.22	2.12
19	**TA29**	1706	1691.1	1678	10.0	314.9	1642	1631.5	1629	6.82	355.2	1.13	2.92
20	**TA30**	1671	1659.1	1644	10.7	372.4	1616	1601.4	1588	6.93	445.3	1.2	3.41
21	**TA31**	1827	1801.4	1785	1.19	289.5	1784	1775	1767	0.17	357.5	1.23	1.01
22	**TA32**	1899	1881.3	1863	5.02	406.4	1837	1823.6	1801	1.52	520.1	1.28	3.33
23	**TA33**	1847	1833.8	1813	1.97	389.4	1824	1815.1	1803	1.41	442.2	1.14	0.55
24	**TA34**	1883	1868.4	1851	1.26	262.7	1854	1843.8	1829	0.05	340.6	1.3	1.19
25	**TA35**	2087	2068.6	2043	1.79	345.9	2026	2019	2011	0.20	436.3	1.26	1.57
26	**TA36**	2012	1888.8	1867	2.64	425.9	1836	1829.9	1825	0.33	534.9	1.26	2.25
27	**TA37**	1901	1875.7	1852	4.57	461.9	1795	1789.4	1779	0.45	568.2	1.23	3.94
28	**TA38**	1749	1732.7	1716	2.57	463.6	1691	1684.7	1676	0.18	602.9	1.3	2.33
29	**TA39**	1867	1854	1837	2.34	299.0	1826	1811.9	1798	0.17	372.9	1.25	2.12
30	**TA40**	1777	1754.8	1736	6.44	469.0	1702	1688.1	1678	2.88	570	1.22	3.34
31	**TA41**	2132	2109.2	2082	10.9	753.1	2049	2035.7	2022	7.78	975.9	1.3	2.88
32	**TA42**	2046	2022.7	1994	6.80	742.6	1983	1971.2	1962	5.09	956.6	1.29	1.6
33	**TA43**	1966	1938.6	1902	5.14	389.1	1923	1897.9	1867	3.21	523.5	1.35	1.84
34	**TA44**	2133	2097.6	2054	6.59	501.6	2024	2008.9	1986	3.06	713.9	1.42	3.31
35	**TA45**	2153	2134.5	2084	4.35	791.5	2037	2015.1	2008	0.55	983.7	1.24	3.65
36	**TA46**	2161	2138.1	2086	7.52	481.9	2063	2045.4	2020	4.12	669.4	1.39	3.16
37	**TA47**	2038	2014.7	1952	9.11	397.5	1942	1928.6	1905	6.48	531.1	1.34	2.41
38	**TA48**	2142	2097.4	2025	5.91	815.1	1987	1971.3	1958	2.40	1048.3	1.29	3.31
39	**TA49**	2157	2101.2	2024	5.69	571.5	2001	1985.4	1976	3.18	771.1	1.35	2.37
40	**TA50**	2101	2064.9	1988	10.0	746.4	1968	1955.1	1935	7.08	1019.6	1.37	2.67
Average Relative Difference					5.68%	Average Relative Difference				3.05%	Average	1.2	2.47%

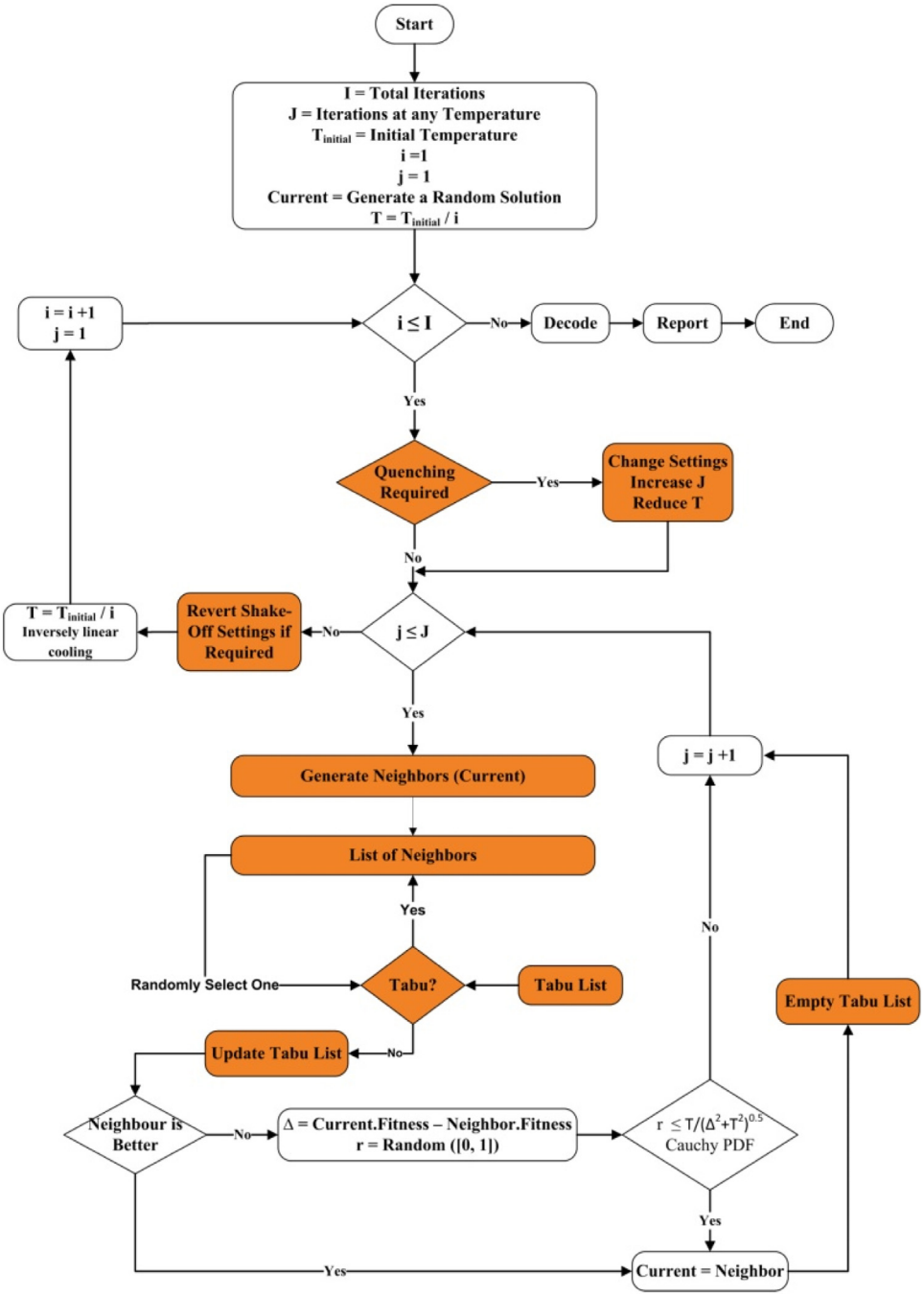

Fig. 8. The flow chart for proposed hybrid method.

The proposed method HFSAQ, which is hybrid of fast simulated annealing FSA, was examined for performance against conventional FSA. For this purpose, benchmark problems were run multiple times by both methods using the same parameters as shown in Table 1 and various performance parameters were recorded: which included worst, best and average makespans, average relative difference (ARD as per equation 4) and average runtime. Benchmark problem LA01 to LA30, FT06, FT10, FT20, ORB01 to ORB10 and ABZ5 to ABZ9 were run 20 times each and their results are summarized in Table 2a. FSA was able to solve 26 out of 48 problems optimally, with 19 zero ARD values. While HFSAQ was able to solve 44 out of 48 problems optimally, with 25 zero ARD values. A zero ARD value means a global optimal solution was found in every run. Overall average ARD values are 1.29% and 0.16% for FSA and HFSAQ respectively. From Table 2a, it is clearly evident that HFSAQ has taken slightly more computational time but more global optimal solutions were found than FSA. Benchmark problems TA11 to TA50 were run 30 times each and their results are summarized in Table 2b. Overall average ARD values are 5.68% and 3.05% for FSA and HFSAQ respectively for these harder problems. Another interesting observation is that even the worst solution of HFSAQ is better than the best solution of FSA for the entire solved problem set, with a slight plenty of, on average, 20% more time. From Table 2a and b it can be concluded that HFSAQ is superior to conventional FSA.

Proposed algorithm provides JSSP schedules in the form of Gantt charts in graphical format as shown in Fig. 9 (for LA24), where

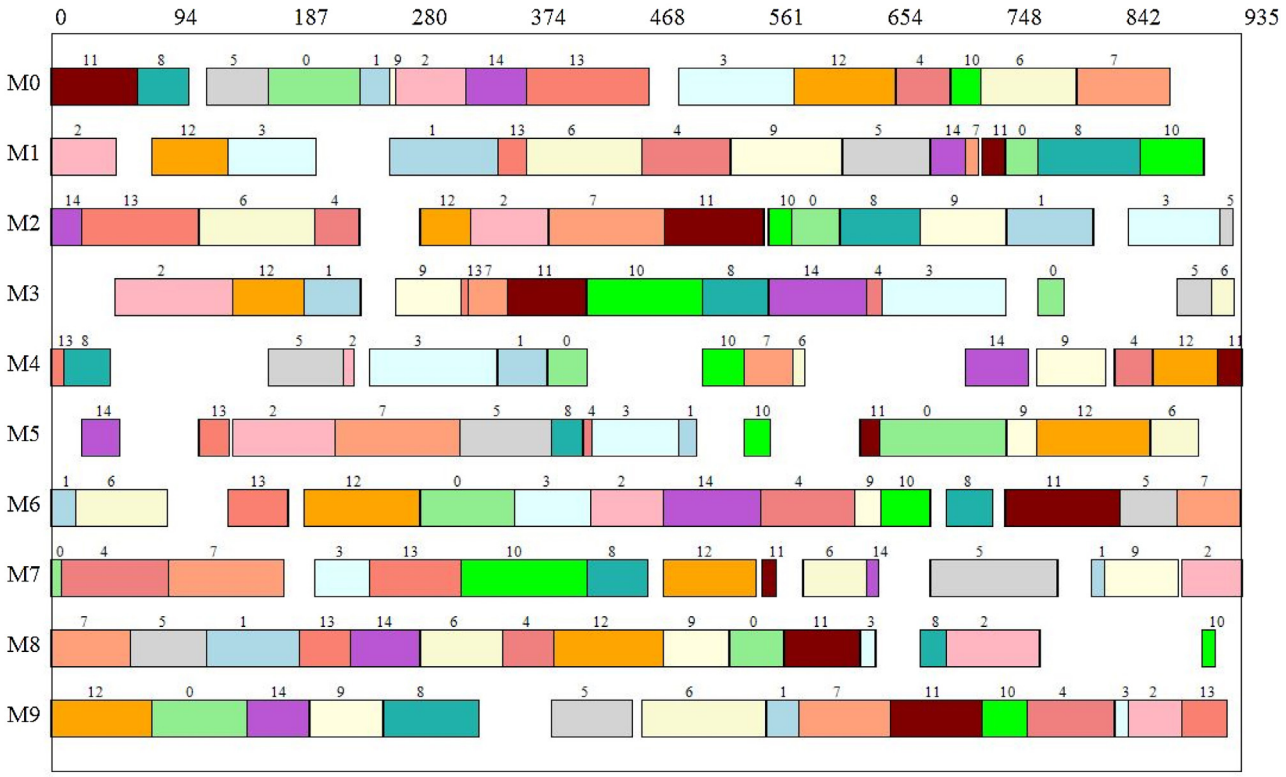

Fig. 9. Solution Gantt chart for instance LA24 (Make span 935).

each job is plotted on horizontal axis against relevant machine. Job numbers are written on the time bars of the individual operations.

After evaluating the performance of HFSAQ and FSA, the proposed method (HFSAQ) is further evaluated by comparing its performance against following 16 other published approaches.

(1) GRASP (Binato et al. [49])
(2) PGRASP (Aiex et al. [50])
(3) HGA (Goncalves et al. [51])
(4) TS (Velmurugan&Selladurai [52])
(5) MA by (Hasan et al. [53])
(6) PaGA by (Asadzadeh and Zamanifar [54])
(7) HPGA by (Gao et al. [32])
(8) HPGA by (Yusof et al. [55])
(9) HGA by (Qing-dao-er-ji and Wang [56])
(10) MP-HGA by (Kalantari and SanieeAbadeh [57])
(11) HBBO by (Wang and Duan [58])
(12) EPPX by (Moin et al. [59])
(13) GATS by (Meeran&Morshed [31])
(14) HAIS by (Qiu and Lau [10])
(15) aLSGA by (Asadzadeh [60])
(16) HSAQ by (Akram and Kamal [37])

Table 3 compare HFSAQ with 16 other approaches for quality of solution, best solutions (BS) for the selected set of problems, for each approach, is reported and compared with proposed algorithm. To make distinction between optimal and non-optimal solutions, non-optimal values are shown in bold face font. At the bottom of the Table 3a, number of instances solved (NIS), number of optimally solved problems and their ratio to NIS is reported for each approach. BKS to NIS ratio for proposed method is 91.7%, while highest ratio of finding BKS to NIS is 95% for Akram and Kamal 2015, but they have only solved a smaller set of 20 problems. Second best BKS to NIS is 90.5%for Meeran and Murshad 2014, and they have solved 42 problems. Only 4 instances LA29, ABZ7 to ABZ9 are not solved to their BKS values by HFSAQ, these instances are considered hard and very few researchers were able to solve these problem optimally or to their respective BKS.

The effectiveness of the proposed method is further demonstrated through comparison of best solutions' relative deviations (BS-RD) with other algorithm as shown in Table 4. BS-RD and average relative deviation (ARD) are calculated by following expressions;

$$\text{BS-RD} = 100 \times (\text{BS} - \text{BKS})/\text{BKS} \tag{4}$$

$$\text{BS-ARD} = \Sigma \text{RD}/\text{NIS} \tag{5}$$

Respective BS-ARD for HFSAQ is calculated, by only considering respective BS for reported problems, for each approach. Last column of Table 4 shows the ratio of BS-ARD of HFSAQ to BS-ARD of other algorithms (OA), a value less than 1 indicates that the proposed algorithm has outperformed the respective benchmark approach. From Table 4 it can be seen that only 2 OAs have surpassed results obtained by HFSAQ, while 14 other benchmark approaches have been outperformed by the proposed HFSAQ algorithm.

The proposed algorithm is further tested on larger Taillard (TA11 to TA50) problems; these instances are considered hardest benchmark problems by research community. Results generated by proposed algorithm are compared with state of art job shop solver algorithms proposed till date, TSSA, proposed by Zhang et al. [44], and iTSAB, proposed by Nowicki and Smutnicki [61]. Table 5a presents results of 30 runs of TA problem with proposed algorithm, best solution, average solution and average runtime of proposed and TSSA algorithms is shown in the table, and only best solutions of iTSAB is provided in Table 5a. Table 5b further summarized the results for set of problem instance TA11 to TA20, TA21 to TA30, TA31 to 40 and TA41 to TA50. For each set average relative difference ARD and average run time is shown for each algorithm.

One of the main factors, which affect execution time of any algorithm, is the target machine. iTSAB algorithm was run on a Pentium III 900 MHz microprocessor machine, TSSA was run on Pentium

Table 3
Comparison of solution quality of HFSAQ with other approaches.

Problem Name	Size	BKS	[49] GRASP BS	[50] PGRA-SP BS	[51] HGA BS	[52] TS BS	[53] MA BS	[54] PaGA BS	[32] MA BS	[55] HPGA BS	[56] HGA BS	[57] MP-HGA BS	[58] HBBO BS	[59] EPPX BS	[31 GATS BS	[10] HAIS BS	[60] aLSGA BS	[37] HSAQ BS	Propos-edAlgori-thm HFSAQ BS
ft06	6 × 6	55	55	55	55	55		55	55		55		55	55	55	55	55		55
ft10	10 × 10	930	**938**	930	930	**935**		**997**	930	**931**	930		930	930	930	930	930		930
ft20	20 × 5	1165	**1169**	1165	1165			**1196**	1165	1165	1165		1165	**1178**	1165	1165	1165		1165
la01	10 × 5	666	666	666	666	666	666	666	666		666	666	666		666	666	666	666	666
la02	10 × 5	655	655	655	655	655	655	655	655	**680**	655	655	655		655		655	655	655
la03	10 × 5	597	**604**	597	597	597	597	**617**	597		597	**621**	597		597		**606**	597	597
la04	10 × 5	590	590	590	590	590	590	**607**	590		590	**602**	590		590		**593**	590	590
la05	10 × 5	593	593	593	593	593	593	593	593		593	593	593		593		593	593	593
la06	15 × 5	926	926	926	926	926	926	926	926		926	926	926		926	926	926	926	926
la07	15 × 5	890	890	890	890	890	890	890	890		890	890	890		890		890	890	890
la08	15 × 5	863	863	863	863	863	863	863	863		863	863	863		863		863	863	863
la09	15 × 5	951	951	951	951	951	951	951	951		951	951	951		951		951	951	951
la10	15 × 5	958	958	958	958	958	958	958	958		958	958	958		958		958	958	958
la11	20 × 5	1222	1222	1222	1222	1222	1222	1223	1222		1222	1222	1222		1222	1222	1222	1222	1222
la12	20 × 5	1039	1039	1039	1039	1039	1039	1039	1039		1039	1039	1039		1039	1039	1039	1039	1039
la13	20 × 5	1150	1150	1150	1150	1150	1150	1150	1150		1150	1150	1150		1150	1150	1150	1150	1150
la14	20 × 5	1292	1292	1292	1292	1292	1292	1292	1292		1292	1292	1292		1292	1292	1292	1292	1292
la15	20 × 5	1207	1207	1207	1207	1207	1207	**1273**	1207		1207	1207	1207		1207		1207	1207	1207
la16	10 × 10	945	**946**	945	945	**946**	945	**994**	945	**947**	945		945		945		**946**	945	945
la17	10 × 10	784	784	784	784	784	784	**793**	784		784		784		784		784	784	784
la18	10 × 10	848	848	848	848	848	848	**860**	848		848		848		848		848	848	848
la19	10 × 10	842	842	842	842	842	842	**873**	842		**844**		842		842		**852**	842	842
la20	10 × 10	902	**907**	902	**907**	902	**907**	**912**	902		**911**		902		902		**907**	**907**	902
la21	15 × 10	1046	**1091**	**1057**	1046		**1079**	**1146**	**1055**	**1067**	1046		1046		**1047**	1046	**1068**		1046
la22	15 × 10	927	**960**	927	**935**		**960**	**1007**	927		**935**		**933**		927		**956**		927
la23	15 × 10	1032	1032	1032	1032		1032	**1033**	1032		1032		1032		1032		1032		1032
la24	15 × 10	935	**978**	**954**	**953**		**959**	**1012**	**940**		**953**		935		935		**966**		935
la25	15 × 10	977	**1028**	**984**	**986**		**991**	**1067**	**984**		**984**		977				**1002**		977
la26	20 × 10	1218	**1271**	1218	1218		1218	**1323**	1218		1218		1218			1218	**1223**		1218
la27	20 × 10	1235	**1320**	**1269**	**1256**		**1286**	**1359**	**1261**		**1236**					**1239**	**1281**		1235
la28	20 × 10	1216	**1293**	**1225**	**1232**		**1286**	**1369**	1216		1216						**1245**		1216
la29	20 × 10	1152	**1293**	**1203**	**1196**		**1221**	**1322**	**1190**	**1154**	**1160**						**1230**		**1164**
la30	20 × 10	1355	**1368**	1355	1355		1355	**1437**	1355		1355						**1355**		1355
orb01	10 × 10	1059				**1064**		**1149**		**1085**			1059	**1077**	1059	1059	**1092**		1059
orb02	10 × 10	888				888		**929**					**919**	**889**	888	888	**984**		888
orb03	10 × 10	1005				1005		**1129**						**1022**	1005	1005	**1029**		1005
orb04	10 × 10	1005				**1011**		**1062**		**1054**				1005	1005	1005	**1016**		1005
orb05	10 × 10	887				887		**936**						**890**	887	887	**901**		887
orb06	10 × 10	1010						**1060**						**1021**	1010	1010	**1028**		1010
orb07	10 × 10	397						**416**						397	397	397	**405**		397
orb08	10 × 10	899						**1010**						899	899	899	**914**		899
orb09	10 × 10	934						**994**						934	934	934	**943**		934
orb10	10 × 10	944												944	944	944			944
abz5	10 × 10	1234	**1238**	1234									**1250**	1234	1234				1234
abz6	10 × 10	943	**947**	943									**948**	943	943				943
abz7	20 × 15	656	**723**	**692**										**683**	**668**				**676**
abz8	20 × 15	665	**729**	**705**										**701**	**682**	**672**			**678**
abz9	20 × 15	678	**758**	**740**										**712**	**679**	**693**			**693**
NIS			38	38	33	27	30	42	33	8	33	15	30	18	42	24	42	20	48
BKS solution found			19	29	26	22	22	14	28	1	26	13	26	9	38	21	20	19	44
Ratio of BKS to NIS			50%	76.3%	78.8%	81.5%	73.3%	33.3%	84.8%	12.5%	68.4%	86.7%	86.7%	50%	90.5%	87.5%	47.6%	95%	91.7%

Non optimal solutions are shown in **bold** font.

Table 4
Comparison of average relative difference of best solutions of HFSAQ with other algorithms.

Algorithm	NIS	BS-ARD		$Ratio = \frac{HFSAQ_{ARD}}{OA_{ARD}}$
		OA (%)	HFSAQ	
[49] GRASP	38	2.207	0.209	0.095
[50] PGRA-SP	38	0.852	0.209	0.246
[51] HGA	33	0.336	0.0316	0.094
[52] TS	27	0.063	0	0
[53] MA	30	0.905	0.035	0.0383
[54] PaGA	42	4.455	0.025	0.006
[32] MA	33	0.228	0.032	0.139
[55] HPGA	8	1.706	0.130	0.076
[56] HGA	33	0.167	0.032	0.189
[57] MP-HGA	15	0.404	0	0
[58] HBBO	30	0.199	0.035	0.175
[59] EPPX	18	1.144	0.442	0.386
[31] GATS	42	0.110	0.189	1.718
[10] HAIS	24	0.150	0.174	1.161
[60] aLSGA	42	1.278	0.025	0.019
[37] HSAQ	20	0.028	0	0

Table 5
(a) Comparison of HFSAQ with TSSA and iTSAB for TA11 to TA50. (b) Comparison of HFSAQ with TSSA and iTSAB for TA problem sets.

Problem	Size	UB(LB)	TSSA			iTSAB	Proposed		
			Best	M_{av}	T_{av}(s)		Best	M_{av}	T_{av}(s)
TA11	20 × 15	1357(1323)	1359	1367.6	260.4	1361	1361	1370.4	178.9
TA12	20 × 15	1367(1351)	1371	1374.3	253.4	1367	1372	1376.9	155.1
TA13	20 × 15	1342(1282)	1342	1355.2	258.6	1342	1342	1360.1	176.2
TA14	20 × 15	1345	1345	1346.7	65.2	1345	1345	1348.7	192.2
TA15	20 × 15	1339(1304)	1339	1348.4	252.7	1340	1340	1350.1	219.9
TA16	20 × 15	1360(1304)	1360	1366.2	253.2	1360	1360	1368.3	208.2
TA17	20 × 15	1462	1464	1472.9	249.4	1462	1464	1473.7	222.7
TA18	20 × 15	1396(1369)	1399	1408.7	231.9	1396	1400	1410.8	178.3
TA19	20 × 15	1332(1304)	1335	1340.6	261.5	1335	1339	1343.2	214.5
TA20	20 × 15	1348(1318)	1350	1356.7	264	1351	1353	1358.0	228.5
TA21	20 × 20	1642(1573)	1644	1650.5	437	1644	1646	1652.7	283.5
TA22	20 × 20	1600(1542)	1600	1606.4	433.5	1600	1606	1607.1	363.3
TA23	20 × 20	1557(1474)	1560	1564.5	429.4	1557	1564	1565.4	451.0
TA24	20 × 20	1644(1606)	1646	1653.2	431.6	1647	1648	1656.8	340.6
TA25	20 × 20	1595(1518)	1597	1607.7	421	1595	1599	1608.8	419.6
TA26	20 × 20	1643(1558)	1647	1654.9	436.2	1645	1647	1659.5	481.6
TA27	20 × 20	1680(1617)	1680	1688.7	447.8	1680	1684	1691.2	401.3
TA28	20 × 20	1603(1591)	1603	1616.6	431.2	1614	1613	1621.9	296.8
TA29	20 × 20	1625(1525)	1627	1630.1	426.2	1625	1629	1631.5	355.2
TA30	20 × 20	1584(1485)	1584	1597.7	436.1	1584	1588	1601.4	445.3
TA31	30 × 15	1764	1764	1765.8	318.3	1764	1767	1775.0	357.5
TA32	30 × 15	1785(1774)	1795	1811.7	613.2	1796	1801	1823.6	520.1
TA33	30 × 15	1791(1778)	1796	1806.7	494.7	1793	1803	1815.1	442.2
TA34	30 × 15	1829(1828)	1831	1831.8	337.9	1829	1829	1843.8	340.6
TA35	30 × 15	2007	2007	2011	129.5	2007	2011	2019.0	436.3
TA36	30 × 15	1819	1819	1820.5	178.1	1819	1825	1829.9	534.9
TA37	30 × 15	1771	1778	1784.4	496.4	1778	1779	1789.4	568.2
TA38	30 × 15	1673	1673	1678.5	333	1673	1676	1684.7	602.9
TA39	30 × 15	1795	1795	1806.6	303.2	1795	1798	1811.9	372.9
TA40	30 × 15	1669(1631)	1676	1684	499.3	1674	1678	1688.1	570.0
TA41	30 × 20	2005(1876)	2018	2028.5	805.5	2018	2022	2035.7	975.9
TA42	30 × 20	1937(1867)	1953	1964.5	805.2	1956	1962	1971.2	956.6
TA43	30 × 20	1848(1809)	1858	1882.6	922.8	1859	1867	1897.9	523.5
TA44	30 × 20	1979(1927)	1983	1998.2	897.9	1984	1986	2008.9	713.9
TA45	30 × 20	2000(1997)	2000	2006.8	805.1	2000	2008	2015.1	983.7
TA46	30 × 20	2004(1940)	2010	2029.2	818.8	2021	2020	2045.4	669.4
TA47	30 × 20	1894(1789)	1903	1918.2	877.4	1903	1905	1928.6	531.1
TA48	30 × 20	1943(1912)	1955	1968.6	802.6	1952	1958	1971.3	1048.3
TA49	30 × 20	1961(1915)	1967	1980	910.9	1968	1976	1985.4	771.1
TA50	30 × 20	1924(1807)	1931	1945.6	811.8	1928	1935	1955.1	1019.6

Problem Group	ARD			Time		
	TSSA[a]	iTSAB[a]	Proposed	TSSA	iTSAB	Proposed
TA 11–20	2.30	2.26	2.39	235	108	197
TA 21–30	4.55	4.56	4.78	433	328	384
TA 31–40	0.55	0.52	0.73	370	341	474
TA 41–50	3.97	4.03	4.29	845	975	819

[a] Values calculated using same LB values(shown in Table 5a) used to calculate ARD for Proposed algorithm.

IV 3.0 GHz microprocessor machine while proposed algorithm was run on Intel Core(2) 2.4 GHz machine. From Table 5a and b, it can be observed that both TSSA and proposed algorithms, even having superior computing machines, have taken more execution time than iTSAB. TSSA has produced better solution quality than iTSAB in comparison to HFSAQ, but with higher execution time than the proposed method. From Table 5b, it can be observed that the solution quality of proposed algorithm is slightly poorer than the TSSA and iTSAB and execution time is poorer than iTSAB and better than TSSA. Therefor it can be concluded that the proposed algorithm does not lag far behind the state of art TSSA and iTSAB.

6. Conclusions and future work

In this paper, a new novel hybrid approach is proposed which combines fast simulated annealing with quenching. In this approach FSA is used for both to perform global search and a way to escape local optima. Quenching is used to perform local search in near vicinity of current solution. The novelty of this approach is two folds one is hybridization of FSA and quenching and the other is its ability to escape local optima even in quenching mode. This approach was tested on 88 well known problems, taken from four groups of benchmark instances. The efficacy of proposed algorithm has been verified by comparing it with other published works of various authors. Proposed algorithm was able to solve all 45 problems optimally in reasonable time, which demonstrate the effectiveness of the proposed method.

For future work, we propose to use some other objective function such as tardiness, lateness and due dates etc. instead of make span.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.asoc.2016.08.037.

References

[1] B.J. Lageweg, J.K. Lenstra, A.H.G. RinnooyKan, Job-shop scheduling by implicit enumeration, Manag. Sci. 24 (4) (1977) 441–450.
[2] H.M. Wagner, An integer linear-programming model for machine scheduling, Naval Res. Logist. Q. 6 (2) (1959) 131–140.
[3] P.B. Luh, D.J. Hoitomt, Scheduling of manufacturing systems using the Lagrangian relaxation technique, IEEE Trans. Autom. Control (1993), http://dx.doi.org/10.1109/9.231461.
[4] M. Held, R.M. Karp, dynamic programming approach to sequencing problems, SIAM J. Appl. Math. 10 (1962) (1962) 196–210.
[5] J. Carlier, E. Pinson, An algorithm for solving the job-shop problem, Manag. Sci. 35 (2) (1989) 164–176.
[6] P. Brucker, B. Jurisch, B. Sievers, A branch-and-bound algorithm for the job-shop scheduling problem, Discrete Appl. Math. 49 (1–3) (1994) 107–127.
[7] F. Glover, H.J. Greenberg, New approaches for heuristic search a bilateral linkage with artificial-intelligence, Eur. J. Oper. Res. 39 (2) (1989) 119–130.
[8] K. Baker, Introduction to Sequencing and Scheduling, Wiley, New York, 1974.
[9] O. Holthaus, C. Rajendran, fficient dispatching rules for scheduling in a job shop, Int. J. Prod. Econ. 48 (1) (1997) 87–105.
[10] X. Qiu, H.Y.K. Lau, An AIS-based hybrid algorithm for static job shop scheduling problem, J. Intell. Manuf. 25 (3) (2014) 489–503.
[11] J. Adams, E. Balas, D. Zawack, he shifting bottleneck procedure for job shop scheduling, Manag. Sci. 34 (3) (1988) 391–401.
[12] S. Dauzere-Peres, J.B. Lasserre, A modified shifting bottleneck procedure for job-shop scheduling, Int. J. Prod. Res. 31 (4) (1993) 923–932.
[13] A. Jain, S. Meeran, state-of-the-art review of job shop scheduling techniques, Eur. J. Oper. Res. 113 (2) (1999) 390–434.
[14] G.R. Weckman, C.V. Ganduri, D.A. Koonce, A neural network job-shop scheduler, J. Intell. Manuf. 19 (2) (2008) 191–201.
[15] S.X. Yang, D.W. Wang, T.Y. Chai, G. Kendall, n improved constraint satisfaction adaptive neural network for job-shop scheduling, J. Sched. 13 (1) (2010) 17–38.
[16] A. Yahyaoui, N. Fnaiech, F. Fnaiech, A suitable initialization procedure for speeding a neural network job-shop scheduling, IEEE Trans. Ind. Electron. (2011), http://dx.doi.org/10.1109/tie.2010.2048290.
[17] R.G. Sache, eural network for solving job shop scheduling problem, J. Comput. Eng. 16 (6) (2014) 18–25.
[18] Y. Li, Y. Chen, A genetic algorithm for job-shop scheduling, J. Softw. 5 (3) (2010) 269–274.
[19] L. Sun, X. Cheng, Y. Liang, Solving job shop scheduling problem using genetic algorithm with penalty function, Int. J. Intell. Inf. Process. 1 (2) (2010) 65–77.
[20] G. Chryssolouris, V. Subramaniam, Dynamic scheduling of manufacturing job shops using genetic algorithms, J. Intell. Manuf. 12 (3) (2001) 281–293.
[21] Z. Lian, W. Lin, Y. Gao, B. Jiao, A discrete particle swarm optimization algorithm for job-shop scheduling problem to maximizing production, Int. J. Innov. Comput. Inf. Control 10 (2) (2014) 729–740.
[22] A. AitZai, B. Benmedjdoub, M. Boudhar, Branch-and-bound and PSO algorithms for no-wait job shop scheduling, J. Intell. Manuf. (2014), http://dx.doi.org/10.1007/s10845-014-0906-7.
[23] M. Amico, M. Trubian, Applying tabu search to the job-shop scheduling problem, Anna. Oper. Res. 41 (3) (1993) 231–252.
[24] F. Geyik, I.H. Cedimoglu, The strategies and parameters of tabu search for job-shop scheduling, J. Intell. Manuf. (2004), http://dx.doi.org/10.1023/b:jims.0000034106.86434.46.
[25] J.M. Laarhoven, E.H.L. Aarts, J.K. Lenstra, Job shop scheduling by simulated annealing, Oper. Res. 40 (1) (1992) 113–125.
[26] S.G. Ponnambalam, N. Jawahar, P. Aravindan, A simulated annealing algorithm for job shop scheduling, Prod. Plann. Control (1999), http://dx.doi.org/10.1080/095372899232597.
[27] S. Song, J. Ren, J. Fan, Improved simulated annealing algorithm used for job shop scheduling problems, Adv. Electr. Eng. Autom. (2012), http://dx.doi.org/10.1007/978-3-642-27951-5_3.
[28] E. Florez, W. Gomez, L. Bautista, n ant colony optimization algorithm for job shop scheduling problem, Int. J. Artif. Intell. Appl. 4 (4) (2013) 53–66.
[29] M. Kurdi, A new hybrid island model genetic algorithm for job shop scheduling problem, Comput. Ind. Eng. (2015), http://dx.doi.org/10.1016/j.cie.2015.07.015.
[30] S. Meeran, M. Morshed, A hybrid genetic tabu search algorithm for solving job shop scheduling problems—a case study, J. Intell. Manuf. (2012), http://dx.doi.org/10.1007/s10845-011-0520-x.
[31] S. Meeran, M.S. Morshed, Evaluation of a hybrid genetic tabu search framework on job shop scheduling benchmark problems, Int. J. Prod. Res. (2014), http://dx.doi.org/10.1080/00207543.2014.911417.
[32] L. Gao, G. Zhang, L. Zhang, X. Li, An efficient memetic algorithm for solving the job shop scheduling problem, Comput. Ind. Eng. (2011), http://dx.doi.org/10.1016/j.cie.2011.01.003.
[33] W. Xia, Z. Wu, A hybrid particle swarm optimization approach for the job shop scheduling problem, Int. J. Adv. Manuf. Technol. (2006), http://dx.doi.org/10.1007/s00170-005-2513-4.
[34] R. Zhang, C. Wu, A hybrid immune simulated annealing algorithm for the job shop scheduling problem, Appl. Soft Comput. (2010), http://dx.doi.org/10.1016/j.asoc.2009.06.008.
[35] T.A. TamilarasiA. kumar, n enhanced genetic algorithm with simulated annealing for job-shop scheduling, Int. J. Eng. Sci. Technol. 2 (1) (2010) 144–151.
[36] R. Thamilselvan, P. Balasubramanie, Integrating genetic algorithm, tabu search and simulated annealing for job shop scheduling problem, Int. J. Comput. Appl. (2012), http://dx.doi.org/10.5120/7348-0283.
[37] K. Akram, K. Kamal, Hybridization of simulated annealing with quenching for job shop scheduling, Proc. Int. Conf. Fluid Power Mechatron. (2015), http://dx.doi.org/10.1109/fpm.2015.7337228.
[38] H. Szu, Fast simulated annealing. Naval research laboratory, code 5709, Washington, D.C, 20375–5000, 1986, 420–425.
[39] J. Beasley, OR-library: distributing test problems by electronic mail, J. Oper. Res. Soc. (1990), http://dx.doi.org/10.2307/2582903.
[40] C. Bierwirth, A generalized permutation approach to job shop scheduling with genetic algorithms, OR Spectr. 17 (1995) 87–92.
[41] E. Nowicki, C. Smutnicki, A fast taboo search algorithm for the job shop problem, Manag. Sci. 42 (6) (1996) 797–813.
[42] M.A. Cruz-Chávez, J. Frausto-Solís, A new algorithm that obtains an approximation of the critical path in the job shop scheduling problem MICAI 2006, Adv. Artif. Intell. (2006) 450–460.
[43] S. Kirkpatrick, C.D. Gelett, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 621–630.
[44] C.Y. Zhang, P. Li, Y. Rao, Z. Guan, A very fast TSSA algorithm for the job shop scheduling problem, Comput. Oper. Res. 35 (2008) 282–294.
[45] H. Fisher, G.L. Thompson, Probabilistic learning combinations of local job-shop scheduling rules, Ind. Sched. (1963) 225–251.
[46] S. Lawrence, Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement), in: Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1984.
[47] D. Applegate, W. Cook, A computational study of job shop problem, ORSA J. Comput. 3 (2) (1991) 149–156.
[48] E. Taillard, Benchmarks for basic scheduling problems, Eur. J. Oper. Res. 64 (1993) 278–285.
[49] S. Binato, W.J. Hery, D.M. Loewenstern, M.G.C. Resende, A GRASP for job shop scheduling, in: C.C. Ribeiro, P. Hansen (Eds.), Essays and Surveys in Metaheuristics, Kluwer Academic Publishers, 2002.
[50] R.M. Aiex, S. Binato, M.G.C. Resende, Parallel GRASP with path-relinking for job shop scheduling, Parallel Comput. 29 (2003) 393–430.
[51] J.F. Goncalves, J.J.D.M. Mendes, M.G.C. Resende, A hybrid genetic algorithm for the job shop scheduling problem, Eur. J. Oper. Res. 167 (2005) 77–95.

[52] P.S. Velmurugan, V. Selladurai, A tabu search algorithm for job shop scheduling problem with industrial scheduling case study, Int. J. Soft Comput. 2 (2007) 531–537.
[53] S. Hasan, R. Sarker, D. Essam, D. Cornforth, Memetic algorithms for solving job-shop scheduling problems, Memet. Comput. (2009), http://dx.doi.org/10.1007/s12293-008-0004-5.
[54] L. Asadzadeh, K. Zamanifar, An agent-based parallel approach for the job shop scheduling problem with genetic algorithms, Math. Comput. Model. (2010), http://dx.doi.org/10.1016/j.mcm.2010.04.019.
[55] R. Yusof, M. Khalid, G.T. Hui, S. Yusof, M.F. Othman, Solving job shop scheduling problem using a hybrid parallel micro genetic algorithm, Appl. Soft Comput. (2011), http://dx.doi.org/10.1016/j.asoc.2011.01.046.
[56] R. Qing-dao-er-ji, Y. Wang, A new hybrid genetic algorithm for job shop scheduling problem, Comput. Oper. Res. (2012), http://dx.doi.org/10.1016/j.cor.2011.12.005.
[57] S. Kalantari, M. SanieeAbadeh, n effective multi-population based hybrid genetic algorithm for job shop scheduling problem, Bull. Electr. Eng. Inform. 2 (1) (2013) 59–64.
[58] X. Wang, H. Duan, A hybrid biogeography-based optimization algorithm for job shop scheduling problem, Comput. Ind. Eng. (2014), http://dx.doi.org/10.1016/j.cie.2014.04.006.
[59] N.H. Moin, O.C. Sin, M. Omar, Hybrid genetic algorithm with multi-parents crossover for job shop scheduling problems, Mathem. Prob. Eng. (2014), http://dx.doi.org/10.1155/2015/210680.
[60] L. Asadzadeh, A local search genetic algorithm for the job shop scheduling problem with intelligent agents, Comput. Ind. Eng. 85 (2015) 376–383.
[61] E. Nowicki, C. Smutnicki, An advanced tabu search algorithm for the job shop problem, J. Sched. 8 (2005) 145–159.

Kashif Akram He has completed his bachelor degree in Industrial and Manufacturing Engineering in 2003, currently he is doing is MS in Mechatronics from National University of Science and Technology Islamabad. He has research interest in application of artificial intelligence and evolutionary algorithm for efficient planning and scheduling of manufacturing activities. He is currently working as a manager production planning and control department of a local manufacturing industry.

Dr. Khurram Kamal He has completed his Ph.D. in Mechatronics from UK. He has research interest in application of Artifiecial intelligence and neural network in manufacturing industry. He is currently teaching at National University of Science and Technology Islamabad as assistant professor.

Alam Zeb He has completed his bachelor degree in Mechanical Engineering in 1999, and master degree in Engineering Management (with specialized in manufacturing management), in 2010. He has research interest in production planning and control, cellular manufacturing and job shop scheduling. He is currently working as a production manager in a local manufacturing industry.