Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)

A New Meta-heuristic Approach for Combinatorial Optimization and Scheduling Problems

Nader Azizi, Saeed Zolfaghari and Ming Liang

***Abstract*—This study presents a new metaheuristic approach that reasonably combines different features of several well-know heuristics. The core component of the proposed algorithm is a simulated annealing that utilizes three types of memories, two short-term memories and one long-term memory. The purpose of the two short-term memories is to guide the search toward good solutions. While the aim of the long term memory is to provide means for the search to escape local optima through increasing the diversification phase in a logical manner. The long-term memory is considered as a population list. In specific circumstances, members of the population might be employed to generate a new population from which a new initial solution for the simulated annealing component is generated. Job shop scheduling problem has been used to test the performance of the proposed method.**

I. Introduction

SINCE the early 1960s the job shop scheduling problem (JSSP) has been benchmark for the quality of newly developed optimization techniques. Among the earliest and the most popular algorithms that have attracted much attention are tabu search [1], simulated annealing [2], and genetic algorithms [3]. Tabu search explores exhaustively the neighborhood of a solution, which is generated by fundamental notion called the *move*. The move is a function that converts a solution into another solution. For each solution, a set of applicable moves is determined. This set is used to generate a group of solutions called the *neighborhood*. Tabu search starts with a feasible solution. The neighborhood of each solution is evaluated to find the one with the lowest cost. The move to the best solution is performed and the new solution becomes an initial solution in the next iteration. In order to prevent cycling, a part of the solution space that has been recently visited is preserved in a short-term memory called *tabu list*. Tabu search, both in its basic form and in hybrid with other heuristics, has been successfully applied to combinatorial optimization problems such as timetabling problem [4]-[5] vehicle routing [6], and job shop scheduling [7]-[8]. Simulated annealing is a stochastic local search technique based on principles of physics. In its basic form, the algorithm begins to search the solution space by selecting a neighbor from the neighborhood of an initial solution. A neighboring solution is generated by a randomized perturbation in the current solution. If the cost value of the candidate solution is lower than that of the current solution, a move from current solution into the candidate solution is made. Otherwise, a *transition probability function* will be used to determine whether to accept or to reject the candidate solution. If the value of the transition probability is greater than a random number (generated from a uniform distribution), then the candidate solution, despite being worse than the current one, is accepted. If the transition to the candidate solution is rejected, another solution from the neighborhood of the current solution will be selected and evaluated.

Genetic algorithms simulate the evolution process of species reproduction [2]. While other heuristics such as simulated annealing and tabu search work with a single solution, genetic algorithms deal with a population of solutions. In the conventional GA, each member of the population is considered as a *chromosome* representing a solution. GA assigns a value to each individual in the population according to a problem-specific objective function. Two chromosomes (individuals) will be selected randomly or with a probability in favor of improved fitness to produce a new generation. The reproduction is then carried out through two fundamental mechanisms. The first one is c*rossover*, which combines parts of the genetic characteristics of two selected parents to produce genetic characteristics of a new individual. The second one is *mutation* by which a spontaneous modification of genetic make-up occurs. The newly created individual is called *child* or *offspring*. This new individual is different from its parents but shares some common characteristics.

In recent years, a number of metaheuristics have been developed. Some examples of those algorithms are Greedy Randomized Adaptive Search Procedure (GRASP) [9]-[10], Adaptive Multi Start [11], Ant System [12], and Adaptive Memory Programming (AMP) [13]. In this study, a new metaheuristic based on several well-known heuristics: simulated annealing, tabu search, and genetic algorithm is presented. The distinctive feature of the proposed method is the use of one long-term and two short-term memories. The two short-term memories can be characterized as negative (inhibitory) and positive (reinforcement) memories. The purpose of these memories is to guide the search of the solution space so that the chance of finding solutions with better quality could be higher. The long-term memory provides means for the search to escape local optima. Job shop scheduling is considered as an example problem to evaluate the performance of the proposed heuristic.

II. Framework of the Proposed Method

Conventional simulated annealing attempts to avoid

Manuscript received October 31, 2006. This work was supported in part by the Natural Science and Engineering Council of Canada.

N. Azizi is a PhD candidate in the Mechanical Engineering Department, University of Ottawa, Canada (e-mail: naziz059@uottawa.ca).

S. Zolfaghari is an associate professor with the Mechanical and Industrial Engineering Department, Ryerson University, Toronto, Canada (corresponding author, phone: 416-979-5000 ext. 7735; fax: 416-979-5265; e-mail: szolfagh@ryerson.ca).

M. Liang is a professor of Mechanical Engineering at the University of Ottawa, Canada (e-mail: mliang@uottawa.ca).

being trapped in a local optimum by sometimes accepting transition corresponding to an increase in cost however; as the search progresses and the transition probability declines, the possibility that the search gets trapped in local optima increases. To alleviate this problem, several methods such as multi-start approaches [14] and adaptive temperature control mechanism [15]-[16] have been suggested in the literature.

Another drawback of the simulated annealing approach is the lack of memory. Some researchers have used a combination of tabu search and simulated annealing to overcome this deficiency [17]. The main advantage of adding a tabu list to a simulated annealing algorithm is that the search would have a memory that provides an inhibitory (negative) feedback to the search, but such memory may still be considered as insufficient as it does not have any positive feedback from the search history. El-bouri *et al*. [18] proposed a framework to combine adaptive memory programming and simulated annealing. The proposed method benefits from both negative (inhibitory) and positive (reinforcement) memories to find solutions with better quality.

To address the above two shortcomings simultaneously, this paper proposes a hybrid algorithm based on simulated annealing, tabu search, and genetic algorithm. The proposed metaheuristic includes five main components: a simulated annealing module, three types of memories, and a genetic algorithm component. The first two memories are short-term that are referred to as the *tabu-elite list* and *seed memory list*. The third memory is a long-term memory and is referred to as *population list*. The mechanism of the search could be described as follows. The algorithm begins the search with an initial feasible solution. Then a neighboring (candidate) solution is generated according to the neighborhood structure of the problem. The candidate solution is evaluated by the simulated annealing module. If the solution is rejected, another solution will be generated and evaluated. In case of acceptance, a condition specific to memories will be verified (e.g., if quality of the candidate solution is better than that of the current solution). If this 'memory' condition is met, first, a small part of the solution or the variable(s), which has been modified to generate the neighboring solution, is temporarily preserved in a tabu-elite list. Then, the entire candidate solution is added to the seed memory list. After updating the memories, the search moves to the candidate solution, and then another solution from the neighborhood of this current solution is generated with respect to the information gathered in the tabu-elite list. If the aforementioned memory condition is not satisfied, then the search moves to the candidate solution without updating or modifying the two memory lists. The above steps are repeated until the tabu-elite list becomes full. Once the iteration is completed, first the best solution from the seed memory list is stored in the population list. Then the same solution is considered as the initial solution for the next iteration of the search (Fig.1). The information stored in both short-term memories will be cleared before starting the next iteration. The above procedure continues until the population list becomes full.

At this stage, a long iteration is said to be completed. Once the population list becomes full, a condition specific to GA component is verified (e.g., if there has been no improvement during the past iteration). If this condition is not met, the search continues by selecting the best solution from the memory list. Before starting the new iteration, all of the memories including the population list are emptied.

If the condition of GA involvement is satisfied, first, a new population is generated using the solutions stored in the population list. If the best member of the new population improves the quality of the best solution found so far, then it will be selected as an initial solution for the next iteration of the algorithm. In case that the best offspring could not improve the quality of the solution, a random number is generated and is compared to the *complement* (or inverse) of the *transition probability* function (CTP function). If this random number is less than the probability function, an offspring from the new population is selected (e.g., randomly) and is considered as an initial solution for the next iteration. Otherwise, the search continues by considering the best solution from the seed memory list as the initial solution for the next iteration.

III. Application of the New Metaheuristic to JSSP

A. Job shop scheduling problem

The deterministic job shop scheduling problem studied in this paper is described as follows. Given a set of jobs and a set of machines, each job consists of a sequence of operations that has to be executed in a specific order. Each operation has to be performed on a given machine without interruption. The objective is to find the schedule that has the minimum makespan (the duration in which all jobs are completed), subject to the following constraints: each machine can handle at most one job at a time, and the operation precedence is respected for each job. The job shop scheduling problem is difficult to solve optimally. This problem is not only NP-hard, it is also considered as one of the most computationally stubborn combinatorial problems. Researchers have employed many techniques to tackle the job shop scheduling problem. Among others, optimization and approximation techniques are very popular. Some instances of the proposed solution techniques for JSSP are as follows. Shifting Bottleneck procedure [19]-[20], Branch and Bound [21]-[23], simulated annealing [24], tabu search [7]-[8], [25], genetic algorithms [26]-[28], hybrid methods [29]-[30], and other techniques including GRASP [31], GRASP with Path-Relinking [10], and Ant colony system [12]. For a comprehensive review of the job shop scheduling problem and its solution techniques, interested readers are referred to [32]-[33].

In order to apply the proposed algorithm to the job shop scheduling problem, the following components are needed to be specified.

1) Encoding Scheme: How to encode a schedule to a solution is a primary and key issue in applying an optimization technique to any scheduling problem. In case

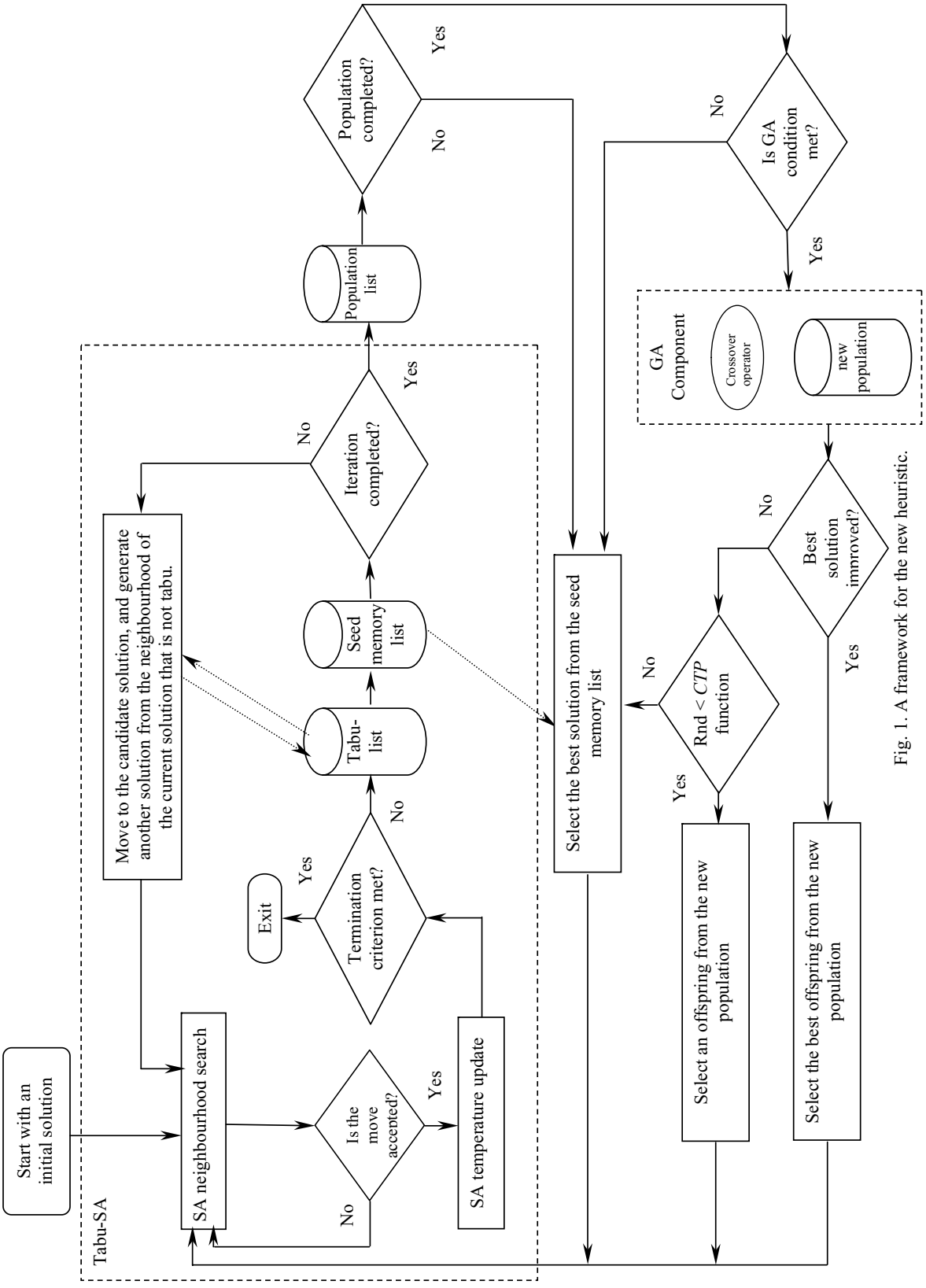


Fig. 1. A framework for the new heuristic.

of job shop scheduling, several encoding scheme such as operation based, job based, machine based, disjunctive graph based, random keys, and binary have been proposed in the literature. Some types of encoding schemes such as binary and permutation may produce an infeasible solution if the solution (generated by these schemes) is subjected to a minor or major perturbation. As a result, a repairing algorithm is usually required to transfer an infeasible solution into a feasible one.

In this study, an operation-based representation proposed by Bierwirth [34] is utilized. This approach uses an un-partitioned permutation of *m*-repetitions of job numbers to represent a solution as a string. Each job is a set of operations that has to be processed on *m* machines. In this form of representation, each job number occurs *m* times in the permutation. Scanning the permutation from left to right, the k^{th} occurrence of a job number refers to the k^{th} operation in the technological sequence of this job. This way, scheduling operations whose technological predecessors have not been scheduled is avoided. For instance, the following string represents a solution for a problem with four jobs and four machines.

4 2 4 1 1 3 2 4 1 4 1 3 3 2 2 3

Each job consists of four operations and is repeated four times. The fifth element of the string is 1. This number refers to the second operation of job 1 because it appears for the second time. Similarly, the ninth and the eleventh elements of the string respectively refer to the third and the fourth operations of job 1.

2) Simulated Annealing Component: In practice, three basic ingredients are needed to apply a simulated annealing algorithm: a cooling schedule, a cost function, and a neighboring solution. In this study, the simulated annealing component embedded in the framework of the proposed heuristic uses the following cooling schedule proposed by Van Laarhoven *et al.* [24].

$$\theta_{i+1} = \frac{\theta_i}{1 + \dfrac{\theta_i \ln(1+\delta)}{3\sigma_i}} \tag{1}$$

where θ_i is the temperature in iteration i, σ_i is the standard deviation of the previously visited solutions and δ is an empirical distance parameter.

The cost function is the makespan of the job shop scheduling problem and a neighboring solution is generated by swapping the position of two non-identical jobs on a string. Given the definition of neighboring solution, size of the neighborhood can be calculated by the following equation:

$$\text{neighbourhood size} = m^2 \left(\sum_{k=1}^{n-1} (n-k) \right) \tag{2}$$

where *m* and *n* denote the number of machines and jobs, respectively.

3) Genetic Algorithm Component: In the proposed heuristic, the genetic algorithm module contains only a random selection mechanism and a crossover operator. Once the population list is completed and if condition applies (i.e., there has been no improvement during the past iteration), a new population is generated by consecutively applying the crossover operator to the randomly selected parents from the current population. Given the solution representation discussed earlier, permutation with *m*-repetitions of the job numbers, precedence preservative crossover (*PPX*) proves to be very efficient [34]. The advantage of this method is that the newly generated individual could be directly decoded as a schedule without any modification. Following this method, a template vector *h* with length *m*×*n* (*m* and *n* denote the number of machines and jobs, respectively) is filled with random elements of set {1,2}. This vector is then used to define the order in which elements are drawn from parent 1 and parent 2. The selected element from one parent is appended to the offspring string and then the corresponding element is deleted from the other parent. This procedure repeats until both parent strings are emptied and the offspring contains all the involved elements.

Fig. 2 shows an illustrative example with four jobs and four machines. P1 and P2 are respectively parent 1 and parent 2 that have been generated randomly. In this example, the first four elements of template vector *h* are 1. Therefore, the first four elements from parent 1 (P1) are appended in offspring string and corresponding elements in parent 2 which are respectively sixth, first, eleventh and third elements are deleted. The second set of the random numbers in vector *h* (fifth to twelfth elements) is 2. Consequently, the first seven elements from the remaining parts in parent 2 (elements 3 2 2 3 1 1 1) are added to the offspring string and then corresponding elements in parent 1 are deleted. The above procedure is repeated until the offspring string contains all the involved elements.

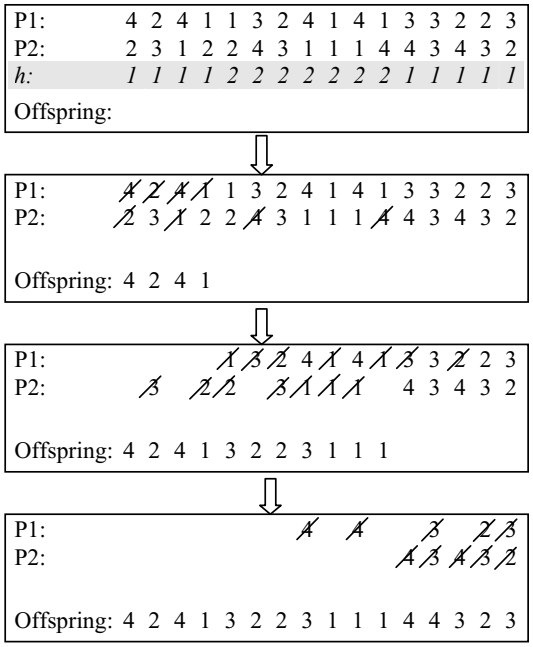


Fig. 2. An illustrative example of the crossover.

B. Proposed meta-heuristic

The proposed algorithm for the job shop scheduling problem can be described as follows. The search begins with an initial solution. Then, a neighboring solution is generated by swapping the position of two non-identical jobs on a string, and is evaluated using the simulated annealing component. If the neighboring solution is rejected, another solution is selected and evaluated. If a move from the current solution to the candidate solution is acceptable (memory condition), then those two jobs whose positions have been exchanged are added into the tabu-elite list; and the candidate solution is added onto the seed memory list. The above steps are repeated until the tabu-elite list reaches its predetermined count. Once the tabu-elite list becomes full, the best solution from the seed memory list is stored in the population list. The same solution is also utilized as the initial solution for the next iteration. At the beginning of the next iteration, both short-term memories are emptied. The aforementioned steps repeated until the population list becomes full (i.e., a longer iteration is completed). It is worthwhile to mention that the size of all three memories is considered to be equal so that they can be controlled by only one parameter that from now on is simply called the *memory size*.

Once the population list is completed, the quality of the best solution found so far is compared with that found at the end of the previous (long) iteration (i.e., the best solution found last time population list reaches the memory size). If the quality of the best solution has been improved, then the best solution from the seed memory is selected and is considered as the initial solution for the next iteration. Then, all of three memories including population list are emptied.

If there has been no improvement during the previous iteration, the condition for GA module is satisfied. Using the current population list and the crossover operator, first, a new population is generated. Then the best member of the new population is selected and evaluated. If the best member of the new population improves the quality of the best solution found so far, then it is considered as the initial solution for the next iteration. In case that the quality of the solution has not been improved by the new population, a random number between zero and one is generated and compared with the complement of the transition probability function (CTP function). If the random number is less than the CTP function, then the new population is scanned and the first member whose cost is different than the cost of the (last) iteration's best solution is selected and it is considered as the initial solution for the next iteration. Following, the temperature is reset at high level. Otherwise, the search moves to the last iteration's best solution (selected from the current seed memory list) without modifying the temperature. In either case, all memories are emptied before starting the new iteration.

Solutions in the new population generally inherit elements from iteration best solutions visited previously. Starting the search from one of these solutions not only guides the search away from local optima, it also increases the chance of improving the best solution in a short period of time. However, because of the structural similarity between these solutions and the best solution found so far, there is also a chance that the search will end up with the same best solution. To reduce the chance of revisiting the same (best) solution, temperature is set to high level at the beginning of the new iteration.

Increasing the level of the temperature (transition probability function) may encourage the search to extend the area of the exploration (diversification). While a low level of the CTP function will prevent the reallocation of the initial solution prematurely (intensification). According to the cooling schedule utilized in this study, the temperature declines from the high of 0.95 with respect to the standard deviation of the previously visited solutions. For this cooling schedule, the CTP function could be defined as:

$$\text{CTP function} = 1 - \theta_i \tag{3}$$

where θ_i is the temperature in iteration i. According to the above equation, setting the temperature at high level any time during the search will automatically retune the *CTP* function to its lowest level. By the same token, as the search continues and the temperature declines, the magnitude of the *CTP* function increases. A higher value of *CTP* function enhances the possibility of success in situations that a decision has to made regarding changing the direction of the search by utilizing a new initial solution (e.g., getting taped in a local minima).

IV. COMPUTATIONAL RESULTS

To evaluate the performance of the proposed heuristic, the algorithm has been coded in Visual Basic and tested using 37 well known classical job shop scheduling problems selected from the literature. The experiments were run on a X86 based PC with 697 MHz CPU. The memory size for small problems (ten jobs and five machines, fifteen jobs and five machines, and twenty jobs and five machines) is set to 10, and for the remaining problems (medium and large) is set to 20. The algorithm has been run several times (five runs for small problems and 20 runs for the rest of the problems) from randomly generated initial solutions.

Table I shows the computational results for benchmark problems. The results presented in this table correspond to the best makespan obtained over several runs and its associated computing time for each problem. Among the 37 instances, the proposed heuristic finds the optimal solution in 29 cases (78.4%) in fairly low computational times. For the other eight instances, a near optimal solution was found (with maximum 0.05 percent deviation from the best known solution in problems ABZ7 and LA29). The algorithm has been run for a fixed computational times ranging from five seconds to seven minutes (420 seconds) depending on the size and complexity of the problem.

In table I, the performance of the proposed metaheuristic is also compared with those of seven well-known algorithms selected from the literature. Tabu search algorithm proposed by Nowicki and Smutnicki [7] is still considered as one of the most efficient techniques for job shop scheduling problem. The other selected methods are two hybrids algorithms proposed by Gonçalves *et al.* [30] and Wang and Zheng [29], simulated

TABLE I
COMPUTATIONAL RESULTS AND COMPARISON WITH OTHER ALGORITHMS

Problem	Size ($n \times m$)	Optimal/BKS	New Heuristic	Gonçalves et al. 2005 [28]	Aiex et al. 2003 [8]	Binato et al. 2002 [29]	Wang and Zheng 2001 [27]	Nowicki and Smutnicki 1996 [5]	Croce et al. 1995 [26]	Van Laarhoven et al. 1992 [22]
MT10	10 × 10	930	930* (168)	930 (292)	930 (10125)	938	930 (44)	930	946 (628)	930 (57772)
MT20	20 × 5	1165	1173 (240)	1165 (204)	1165 (209160)	1169	1165 (90)	1165	1178 (675)	1165 (62759)
ABZ5	10 × 10	1234	1234* (91)	-	1234 (2530)	1238	-	1234	-	-
ABZ6	10 × 10	943	943* (38)	-	943 (678)	947	-	943	-	-
ABZ7	20 × 15	656	690 (373)	-	692 (583000)	723	-	-	-	-
ABZ8	20 × 15	627-670	695 (249)	-	705 (497800)	729	-	-	-	-
LA01	10 × 5	666	666* (2)	666 (37)	666 (<1)	666	666 (6)	666	666 (282)	666 (20)
LA02	10 × 5	655	655* (3)	655 (51)	655 (10.9)	655	-	655	-	655 (24)
LA03	10 × 5	597	597* (13)	597 (39)	597 (29.5)	604	-	597	-	606 (129)
LA04	10 × 5	590	590* (1)	590 (42)	590 (15.7)	590	-	590	-	590 (121)
LA05	10 × 5	593	593* (<1)	593 (32)	593 (<1)	593	-	593	-	593 (5)
LA06	15 × 5	926	926* (<1)	926 (99)	926 (<1)	926	926 (12)	926	926 (473)	926 (16)
LA07	15 × 5	890	890* (1)	890 (86)	890 (<1)	890	-	890	-	890 (66)
LA08	15 × 5	863	863* (<1)	863 (99)	863 (1.6)	863	-	863	-	863 (16)
LA09	15 × 5	951	951* (<1)	951 (94)	951 (<1)	951	-	951	-	951 (13)
LA10	15 × 5	958	958* (<1)	958 (91)	958 (1.15)	958	-	958	-	958 (14)
LA11	20 × 5	1222	1222* (1)	1222 (197)	1222 (<1)	1222	1222 (26)	1222	1222 (717)	1222 (32)
LA12	20 × 5	1039	1039* (<1)	1039 (201)	1039 (<1)	1039	-	1039	-	1039 (34)
LA13	20 × 5	1150	1150* (<1)	1150 (189)	1150 (<1)	1150	-	1150	-	1150 (32)
LA14	20 × 5	1292	1292* (<1)	1292 (187)	1292 (<1)	1292	-	1292	-	1292 (27)
LA15	20 × 5	1207	1207* (3)	1207 (187)	1207 (24.5)	1207	-	1207	-	1207 (34)
LA16	10 × 10	945	945* (68)	945 (232)	945 (2951)	946	945 (29)	945	979 (637)	956 (686)
LA17	10 × 10	784	784* (30)	784 (216)	784 (65.8)	784	-	784	-	784 (112)
LA18	10 × 10	848	848* (84)	848 (219)	848 (453.5)	848	-	848	-	861 (112)
LA19	10 × 10	842	842* (36)	842 (235)	842 (302.8)	842	-	842	-	848 (830)
LA20	10 × 10	902	909 (289)	907 (235)	902 (27710)	907	-	902	-	902 (667)
LA21	15 × 10	1040-1053	1055 (264)	1046 (602)	1057 (351000)	1091	1058 (184)	1047	1097 (1062)	1063 (1991)
LA22	15 × 10	927	937 (149)	935 (594)	927 (89700)	960	-	927	-	938 (2163)
LA23	15 × 10	1032	1032* (92)	1032 (598)	1032 (393.9)	1032	-	1032	-	1032 (275)
LA25	15 × 10	977	985 (399)	986 (609)	984 (105280)	1028	-	977	-	992 (2133)
LA26	20 × 10	1218	1218* (125)	1218 (1388)	1218 (22225)	1271	1218 (418)	1218	1231 (1545)	1218 (4342)
LA29	20 × 10	1120-1195	1219 (259)	1196 (1350)	1203 (308500)	1293	-	1160	-	1218 (4408)
LA30	20 × 10	1355	1355* (91)	1355 (1260)	1355 (22830)	1368	-	1355	-	1355 (3956)
LA31	30 × 10	1784	1784*(37)	1784 (3745)	1784 (2676)	1784	1784 (546)	1784	1784 (2762)	1784 (1517)
LA33	30 × 10	1719	1719*(25)	1719 (3637)	1719 (875.3)	1719	-	1719	-	1719 (1880)
LA34	30 × 10	1721	1721* (128)	1721 (3615)	1721 (4016.5)	1753	-	1721	-	1721 (1886)
LA35	30 × 10	1888	1888* (42)	1888 (3716)	1888 (3483.2)	1888	-	1888	-	1888 (434)

BKS: Best Known Solution; *: Optimal solution

annealing [24], genetic algorithm [28], GRASP [31], and GRASP with path-relinking [10].

In Table I, numbers in parentheses represent the time (sec.) at which the best solution has been found. The results presented in this table clearly indicate that the proposed heuristic outperforms other algorithms in terms of solution quality and/or computational time except for the state-of-the-art tabu search algorithm proposed by Nowicki and Smutnicki [7]. For instance, a comparison of the results obtained by our algorithm with those provided by Gonçalves et al. [30] indicates that for 27 (out of 33) test problems both algorithms have obtained the same results. However, the proposed heuristic without any exceptions, significantly outperforms the other algorithm in terms of computational times.

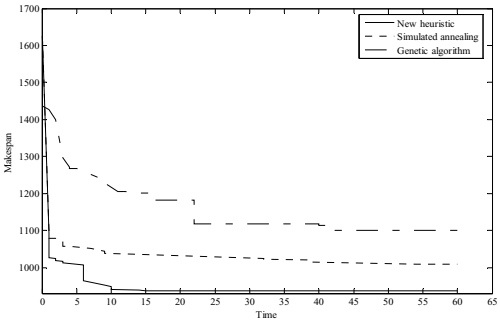
Furthermore, the convergence behavior of the method is studied and compared with conventional simulated annealing and genetic algorithm. All three algorithms use the same encoding scheme, neighborhood structure, and cost function. In GA algorithm, the size of the population is set to 100 and the probability of crossover and mutation are respectively set at 0.85 and 0.1. The performance of the methods on the test problems MT10, LA21, LA26, and LA34 is depicted in Fig. 3(a)-(d).

The simulated annealing algorithm and the new heuristic have been run from the same initial solutions in four test problems. The cost values of the initial solutions are 1326, 1923, 2221, and 3190, respectively. Initial populations for the genetic algorithm have been generated randomly. The cost values of the best member of the populations for each problem are 1436, 1654, 2035, and 2852, respectively.

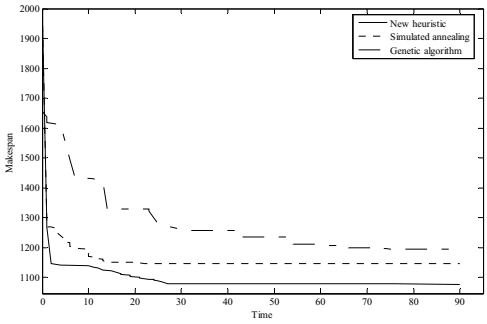
V. Conclusion

This paper presents a new and efficient heuristic by reasonably combining different features of several heuristics: simulated annealing, genetic algorithm, and tabu search. In contrast with other hybrid algorithms, the core component of the proposed heuristic is a simulated annealing that benefits from two short-term memories. information about the past iteration-best solutions is preserved in a long term memory called population list. A genetic crossover operator is used to produce new population using the solutions stored in the third memory in due course. If condition applies, one of the newly produced offspring might be used as an initial solution for the subsequent iteration.

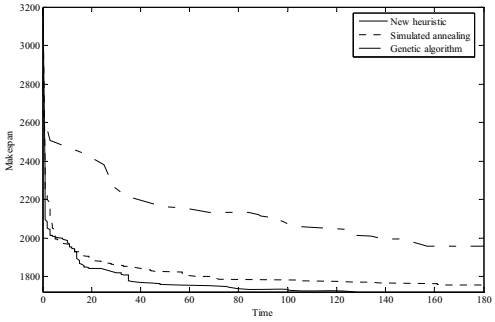
The performance of the proposed hybrid algorithm has been tested using 37 well known classical job shop scheduling problems. The algorithm finds the optimal solution for 29 instances (78.4%). For the remaining problems, a near optimal solution was obtained. The performance of the algorithm is also evaluated in comparison with the results obtained by seven other algorithms including exact methods, general techniques and hybrid algorithms. Comparison of the results clearly indicates that our algorithm can find better solutions in some cases (e.g., [29], [26], and [22]) and it is computationally more efficient than other algorithms



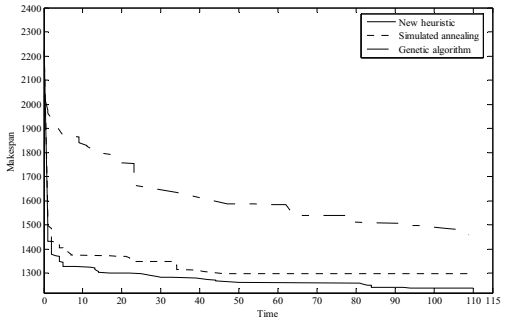
(a) Test problem MT10.



(b) Test problem LA21.



(c) Test problem LA34.



(d) Test problem LA26.

Fig. 3. Convergence comparison of the proposed heuristic with conventional simulated annealing and genetic algorithm.

except for the tabu search algorithm proposed by Nowicki and Smutnicki [5].

The proposed algorithm, due to its generality, can be easily applied to other optimization problems. For the case of job shop scheduling, the quality of the solutions and the computational efficiency of the algorithm can be further improved by using problem-specific neighborhood structure and more efficient operators.

REFERENCES

[1] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers and Operations Research*, vol.13, pp. 533-549, 1986.

[2] S. Kirkpatrick, C.D. Jr. Gelatt, M.P. Vecchi, "*Optimization by simulated annealing*," Science, vol. 220,pp. 671-680,1983.

[3] J.H. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michgan Press, Ann Arbor, MI, 1975.

[4] H. G. Santos, L. S. Ochi , M. J.F. Souza," A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem," *in Proc. Practice and Theory of Automated Timetabling*, *PATAT*, 2004, pp.343-359.

[5] T M. Rahoual, R. Saad, "Solving timetabling problems by hybridizing genetic algorithms and tabu search," *in Proc. Practice and Theory of Automated Timetabling*, *PATAT*, 2006, pp.467-472.

[6] E.D. Taillard, P. Badeau, M. Gendreau, F. Guertin, J.Y. Potvin, "A tabu search heuristic for the vehicle routing problem with soft time windows," *Transportation Science*, vol. 31, pp. 170-186, 1997.

[7] E. Nowicki, C. Smutnicki, "A fast tabu search algorithm for the job shop problem*," Management Science*, vol. 42, pp. 797-813, 1996.

[8] M. Dell'Amico, M. Trubian, "Applying tabu search to the job shop scheduling problem," *Annals of Operations Research*, vol. 41, pp. 231-252, 1993.

[9] T. Feo, M. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 16, pp. 109-133, 1995.

[10] R.M. Aiex, S. Binato, M.G.C. Resende, "Parallel GRASP with path relinking for job shop scheduling," *Parallel Computing*, vol. 29, pp. 393-430, 2003.

[11] K.D. Boese, A.B. Kahng, S. Muddu "A new adaptive multi-start technique for combinatorial global optimizations," *Operations Research Letters*, vol. 16, pp. 101-113, 1994.

[12] M. Dorigo, L.M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem,". IEEE *Trans*. *Evolutionary Computation* 1, 53-66, 1997.

[13] E.D. Taillard, L.M. Gambardella, M. Gendreau, J.Y. Potvin, "Adaptive memory programming: A unified view of metaheuristics," *European Journal of Operational Research*, vol. 135, pp. 1-16, 2001.

[14] E.H.L. Aarts, P.J.M. van Laarhoven, J.K. Lenstra, N.L.J. Ulder, "A computational study of local search algorithms for job shop scheduling," *ORSA Journal on Computing*, vol. 6, pp. 118-125, 1994.

[15] M. Kolonko, "Some new results on simulated annealing applied to the job shop scheduling problem," *European Journal of Operations Research*, vol. 113, pp. 123-136, 1999.

[16] N. Azizi, S. Zolfaghari, "Adaptive temperature control for simulated annealing: a comparative study," Computers and Operations Research, vol. 31, pp. 2439-2451, 2004.

[17] I.H. Osman," Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Annals of Operations Research*, vol. 41, pp. 421-451, 1993.

[18] A. El-Bouri, N. Azizi, S. Zolfaghari, " A comparative study of new metaheuristics based on simulated annealing and adaptive memory programming, *European Journal of Operations Research*, in press.

[19] J. Adams, E. Balas, D. Zawack, "The shifting bottleneck procedure for job shop scheduling." *Management Science*, vol. 34, pp. 391-401, 1988.

[20] S.J. Mason, J.W. Fowler, W. Matthew Carlyle, "A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops," *Journal of Scheduling*, vol. 5, pp. 247-262, 2002.

[21] J. Carlier, E. Pinson, "An algorithm for solving the job shop problem," *Management Science*, vol. 35, pp. 164-176, 1989.

[22] D. Applegate, W. Cook, "A computational study of job shop scheduling problem." *ORSA Journal on Computing*, vol. 3, pp.149-156, 1991.

[23] W. Brinkkotter, P. Brucker, "Solving open benchmark instances for the job-shop problem by parallel head-tail adjustments," *Journal of Scheduling*, vol. 4, pp. 53-64, 2001.

[24] P.J.M. Van Laarhoven, E.H.L. Aarts, J.K. Lenstra, "Job shop scheduling by simulated annealing," *Operation Research*, vol. 40, pp. 113-126, 1992.

[25] E.D. Taillard, "Parallel tabu search techniques for the job shop scheduling problem," *ORSA Journal on Computing*, vol. 6, pp. 108-117, 1994.

[26] D. C. Mattfeld, C. Bierwirth, "An efficient genetic algorithm for job shop scheduling with tardiness objectives," *European Journal of Operational Research*, vol. 155, pp. 616 –630, 2004.

[27] T. Yamada, R. Nakano, " Genetic algorithms for job-shop scheduling problems," in *Proc. Modern Heuristic for Decision Support*, *UNICOM seminar*, 1997, pp.67-81.

[28] F. Della Croce, R. Tadei, G. Volta, "A genetic algorithm for the job shop problem," *Computers & Operations Research*, vol. 22, pp. 15-24, 1995.

[29] L. Wang, D.Z. Zheng, "An effective hybrid optimization strategy for job-shop scheduling problems," *Computers & Operations Research*, vol. 28, pp. 585-596, 2001.

[30] J.F. Gonçalves, J.J.M. Mendes, M.G.C. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *European Journal of Operational Research*, vol. 167, pp. 77-95, 2005.

[31] S. Binato, W.J. Hery, D. Loewenstern, M.G.C. Resende. *A GRASP for job shop scheduling*. Kluwer Academic Publishers, 2001, pp. 59-79.

[32] A.S. Jain, S. Meeran, "Deterministic job-shop scheduling: Past, present and future," *European Journal of Operational Research*, vol. 113, pp. 390-434, 1999.

[33] R.J.M. Vaessens, E.H.L. Aarts, J.K. Lenstra, "Job shop scheduling by local search," *ORSA Journal on Computing*, vol. 8, pp. 302-317, 1996.

[34] C. Bierwirth, D.C. Mattfeld, H. Kopfer, *On Permutation Representations for Scheduling Problems*. Springer-Verlag, 1996, pp. 310–318.