A Collaborative Search Strategy to Solve Combinatorial Optimization and Scheduling Problems

Nader Azizi [a], Saeed Zolfaghari [b] and Ming Liang [a]
[a] *Department of Mechanical Engineering, University of Ottawa Ottawa, Ontario, Canada*
[b] *Department of Mechanical and Industrial Engineering, Ryerson University Toronto, Ontario, Canada*

1. Introduction

Since the creation of operations research as a discipline, a continued interest has been in the development of heuristics or approximation algorithms to solve complex combinatorial optimization problems. As many problems have been proven computationally intractable, the heuristic approaches become increasingly important in solving various large practical problems.
The most popular heuristics that have been widely studied in literature include Simulated Annealing (SA) (Kirkpatrick 1983), Tabu Search (TS) (Glover 1986), and Genetic Algorithm (GA) (Holland 1975). Simulated annealing is a stochastic search method that explores the solution space using a hill climbing process. Tabu search, on the other hand, is a deterministic search algorithm that attempts exhaustive exploration of the neighbourhood of a solution. In contrast to the local search algorithms such as SA and TS that work with one feasible solution in each iteration, GA employs a population of solutions and is capable of both local and global search in the solution space.
Despite their successes in solving many combinatorial problems, these algorithms have some limitations. For instance, SA can be easily trapped in a local optimum or may require excessive computing time to find a reasonable solution. To successfully implement a tabu search algorithm, one requires a good knowledge about the problem and its solution space to define an efficient neighbourhood structure. Genetic algorithms often converge to a local optimum prematurely and their success often relies on the efficiency of the adopted operators.
Due to the deficiencies of the conventional heuristics and ever increasing demand for more efficient search algorithms, researchers are exploring two main options, developing new methods such as nature inspired metaheuristics and investigating hybrid algorithms.
In recent years, a number of metaheuristics have been developed. Examples include the Greedy Randomized Adaptive Search Procedure (GRASP) (Feo & Resende, 1995; Aiex *et al.*, 2003), Adaptive Multi Start (Boese *et al.*, 1994), Adaptive Memory Programming (AMP)

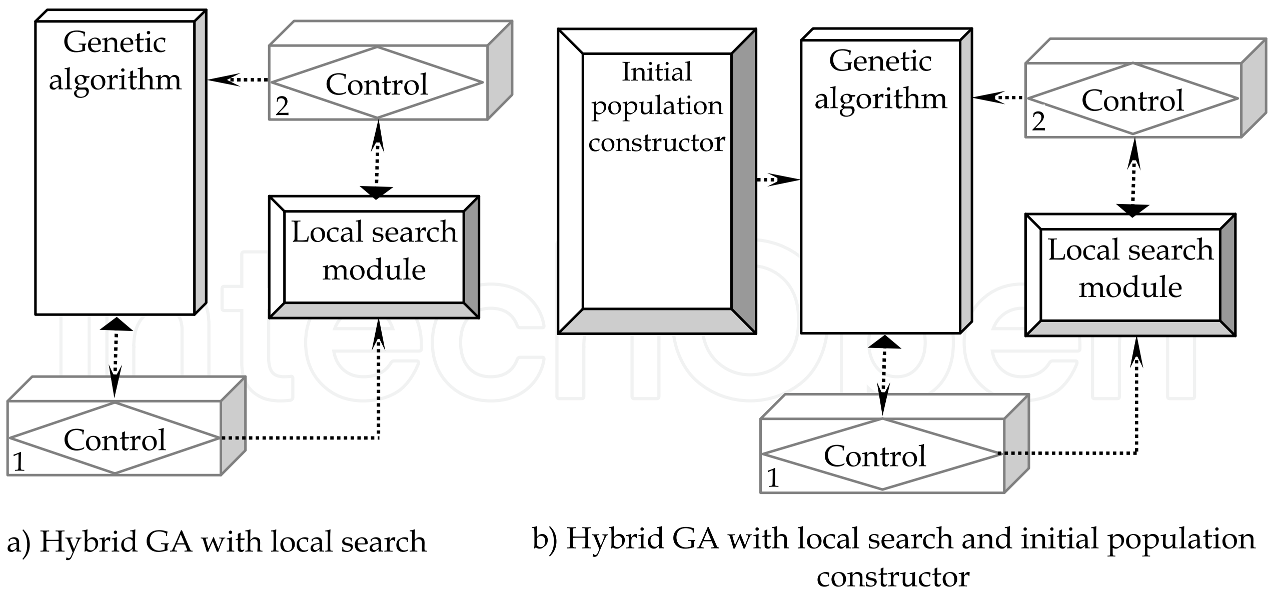
(Taillard *et al*., 2001), Ant System (AS) (Dorigo & Gambardella, 1997), and Particle Swarm Optimization (PSO) (Kennedy & Eberhart 1995).
Another important research domain in heuristics is *hybridization* by which a search algorithm such as GA is used in conjunction with one or more other techniques, e.g., simulated annealing and/or tabu search. Because of the complementary properties of each algorithm, hybrid approaches often outperform each individual method operating alone. However, they still require excessive computational effort.
In view of the above, this chapter presents a Collaborative Search Algorithm (CSA) for combinatorial optimization problems. The proposed approach contains two independent search algorithms, an enhanced simulated annealing with memory and a blockage removal feature (called Tabu-SA component hereafter), and a genetic algorithm (i.e., GA component). The two algorithms exchange information while consecutively run to solve a problem. The Tabu-SA algorithm utilizes a short-term memory, i.e., tabu list to avoid revisiting solutions temporarily. The purpose of the blockage removal feature in the Tabu-SA component is to resolve deadlock situation in which a (acceptable) solution is hard to find in the neighbourhood of a solution. The GA component employs two evolutionary operators, crossover and mutation operators, and a long-term memory, i.e., population list to conduct global search. The population list is constantly updated by the two algorithms, Tabu-SA and GA, as the search progresses.
The reminder of this chapter is organized as follow. In section 2, a review of the two popular hybrid algorithms, hybrid GA and hybrid SA, is presented. Section 3 describes the proposed collaborative search algorithm (CSA). In section 4, more details of its application to flow shop scheduling problem is presented. Computational results and the comparisons with other approaches followed by conclusion are presented in sections 5 and 6.

2. Review of Hybrid GA and SA Algorithms

Depending on which algorithm is selected to be the core component of the hybrid algorithms, the hybrid algorithms could be categorized as hybrid SA or hybrid GA. In a hybrid SA, the driving module of the algorithm is a simulated annealing while in a hybrid GA the driving component is a genetic algorithm.

2.1 Hybrid genetic algorithms

The popular forms of hybrid GAs are constructed by: a) including a constructive heuristic to generate one or more members of the initial population (e.g., Reeves, 1995); b) incorporating a local search heuristic to improve the quality of the solutions built by the crossover operator (e.g., Gonçalves *et al*., 2005); and c) by combination of the former two (a and b) strategies (e.g., Wang & Zheng, 2003; Ruiz *et al*., 2006; and Kolonko, 1999). Figure 1 illustrates the framework of the two popular hybrid genetic algorithms.
Reeves (1995) applied a genetic algorithm to the permutation flow shop scheduling problem. To generate the initial population, the author used the NEH (Newaz *et al*., 1983) heuristic to generate the first member of the population and the rest of the population were generated randomly.
Wang and Zheng (2003) proposed a hybrid GA for the flow shop scheduling problem. In the hybrid algorithm, the NEH algorithm (Newaz *et al*., 1983) is utilized to generate the first member of the initial population while, the rest of the individuals are generated randomly.



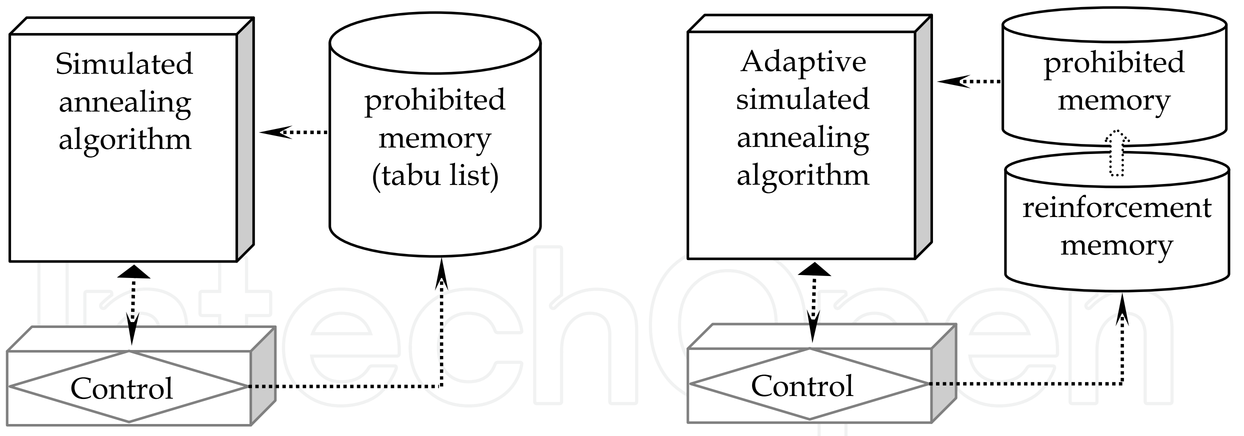
a) Hybrid GA with local search b) Hybrid GA with local search and initial population constructor

Fig. 1. Hybrid genetic algorithm

To improve the performance of their algorithm, Wang and Zhang (2003) replaced the simple mutation operator of the plain GA with a simulated annealing module. Gonçalves *et al.*, (2005) developed a hybrid GA for job shop scheduling problem. In Gonçalves *et al.*, (2005), a local search is used to improve the quality of all individuals built by the crossover operator. Instead of using mutation operator, Gonçalves *et al.*, (2005) replaced 20 percent of the population with randomly generated schedules (i.e., chromosomes).
Ruiz *et al.*, (2006) also proposed a hybrid GA for the flow shop scheduling problem. In their method, the entire initial population is generated using a modified NEH (Newaz *et al.*, 1983) algorithm. Furthermore, each time when a new population is entirely set up, the algorithm enhances the generation by applying a local search to the best member of the new population. Kolonko (1999) proposed a hybrid GA algorithm in which members of population are independent SA runs. Each member of the population is enhanced by a SA module as long as the number of trials without improvement is less than a predetermined value. To generate a new population, two individuals are selected randomly and their schedules are crossed using a special type of crossover operator. The hybrid algorithm was favourably tested using several job shop scheduling problems.

2.2 Hybrid simulated annealing algorithms

Two common forms of hybrid SAs that have been reported in literature are built by: a) adding prohibited memory (i.e., tabu list) to simulated annealing algorithm (e.g., Osman, 1993; Zolfaghari & Liang, 1999; Azizi & Zolfaghari, 2004); and b) addition of both prohibited and reinforcement memories (e.g., El-Bouri *et al.*, 2007). Recently a new hybrid simulated annealing has been also reported in the literature that combines three algorithms, simulated annealing, genetic algorithm, and tabu search (Azizi *et al.*, 2009a and Azizi *et al.*,2009b).
Osman (1993) compared the performance of several heuristics including a hybrid simulated annealing that utilizes a tabu list. Osman (1993) concluded that the hybrid algorithm outperforms the conventional simulated annealing both in terms of solutions quality and


Simulated annealing algorithm
prohibited memory (tabu list)
Control
Adaptive simulated annealing algorithm
prohibited memory
reinforcement memory
Control

a) Tabu-simulated annealing algorithm b) Adaptive tabu-simulated annealing algorithm

Fig. 2. Hybrid simulated annealing with memory

computational time. Zolfaghari and Liang (1999) proposed a hybrid tabu-simulated annealing approach to solve the group scheduling problem. The performance of the hybrid method was tested and favourably compared with two other algorithms using tabu search and simulated annealing alone. Azizi and Zolfaghari (2004) proposed an *adaptive* simulated annealing method complemented with a tabu list. The performance of the hybrid algorithm was evaluated and compared with conventional simulated annealing using classical job shop scheduling problems. Using statistical analysis, Azizi and Zolfaghari (2004) showed that the adaptive tabu-SA algorithm outperforms a stand-alone simulated annealing algorithm.

El-Bouri *et al.*, (2007) proposed four versions of a new memory-based heuristic based on AMP and simulated annealing. The main characteristic of the proposed methods is the use of two short-term memories. The first memory is a tabu list while the second is called seed memory list that keeps tracks of the best solution visited during the last iteration. El-Bouri *et al.*, (2007) showed that the simultaneous use of prohibited and reinforcement memories in simulated annealing could significantly improve the search performance.

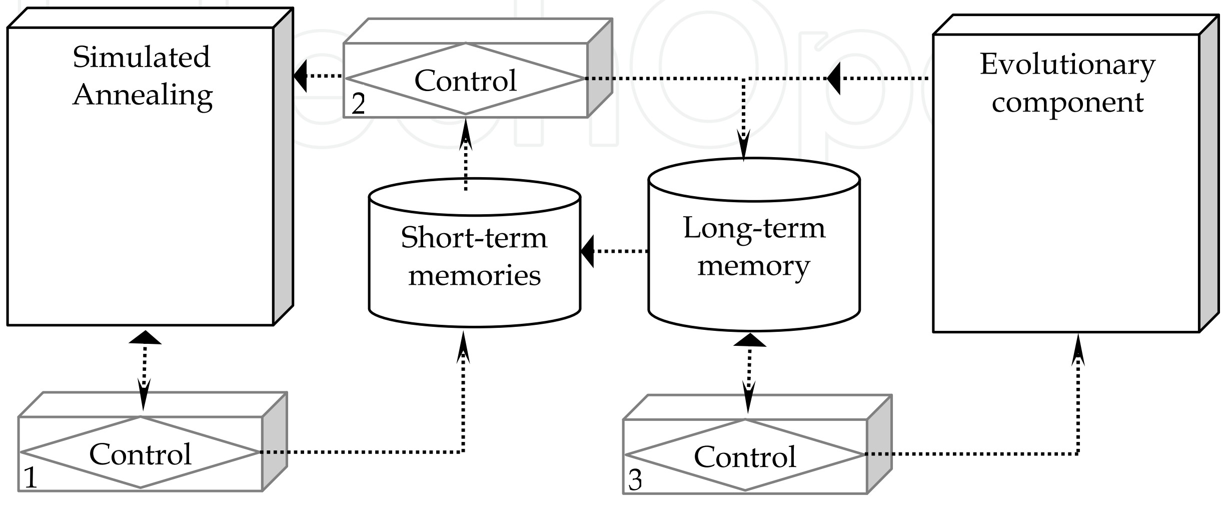


Fig. 3. The generic framework of the SAMED algorithm

Azizi *et al*., (2009a) proposed a generic framework of a new metaheuristic, SAMED, that combines different features of several search algorithms. The framework contains several components including a simulated annealing module, three types of memories, and an evolutionary operator. Based on the generic framework, Azizi *et al*., (2009a & 2009b) developed several search algorithms to investigate the application of the proposed framework on two popular scheduling problems, job shop and flow shop scheduling problems. Azizi *et al*., (2009a & 2009b) showed that the new hybrid algorithms surpass the performance of the conventional heuristics as well as several hybrid genetic algorithms in both applications. The framework of the SAMED algorithm is presented in Figure 3. In the figure, control 1 verifies if the current solution is accepted, and controls 2 and 3 check if the short and long iterations are completed respectively.
For the case of job shop scheduling, Azizi *et al*. (2009a) developed another version of the new metaheuristic that includes two new components. The first component features a problem-specific local search that explores the neighbouring solutions on a critical path of a job shop scheduling solution. The second component is for blockage removal to resolve possible deadlock situations that may occur during the search (Figure 4). The algorithm is named SAMED-LB. Azizi *et al*. (2009a) showed that, in the majority of tested benchmark problems, the addition of the new components has significantly improved the computational efficiency.

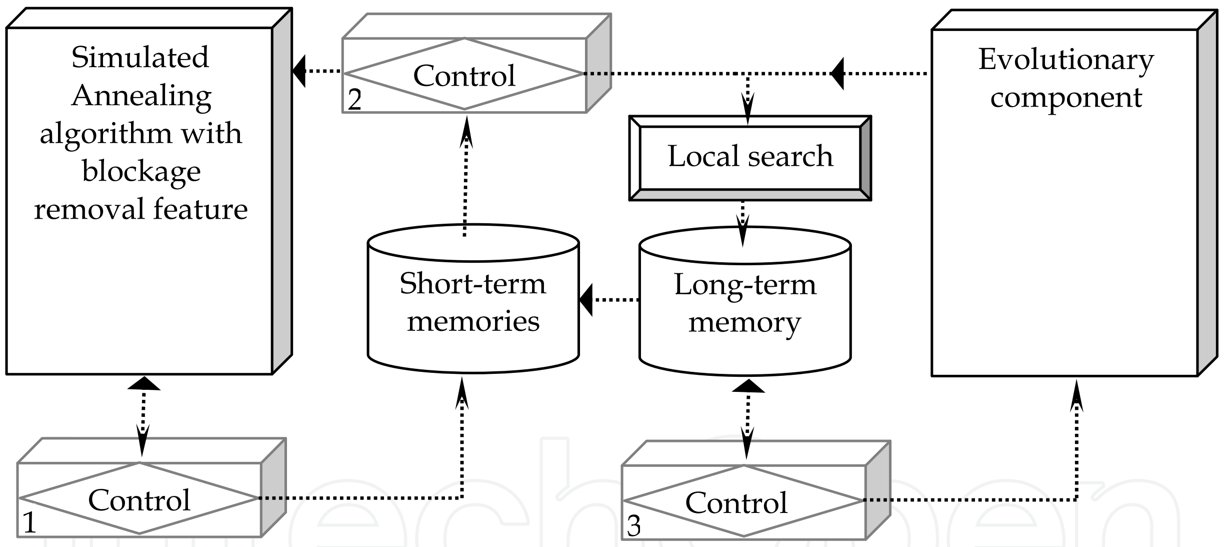


Fig. 4. The generic framework of the SAMED-LB algorithm

3. The Collaborative Search Algorithm

The CSA contains two main components, a Tabu-SA algorithm with blockage removal capability (Azizi *et al*., 2009b) and a genetic algorithm. These two components exchange information (i.e., solutions) via a population list while consecutively run to solve a problem. In the CSA, the original population is generated randomly or by means of other heuristics. The best member of the population is selected as the initial solution for the Tabu-SA module. The Tabu-SA module begins searching the solution space with an initial solution.

Each time a solution is accepted, the quality of the solution is first compared to the overall best solution. Only if the solution improves the quality of the overall best solution, it is added to the population list. Then part of the solution that has been modified to generate the neighbouring solution is added to a tabu list. The completion of the tabu list marks the end of a short iteration. During a short iteration, the Tabu-SA also keeps track of the best solution visited during the iteration. At the end of the iteration, the tabu list is emptied and the best solution of the current iteration (if it is not the current solution) is selected as the initial solution for the next iteration. The above steps are repeated for a number of predetermined short iterations, i.e., a long iteration. Once a long iteration is completed, the performance of the search is evaluated by comparing the current (long) iteration best solution with that at the beginning of the past iteration. In the case of improvement, the Tabu-SA will continue the search; otherwise, the genetic algorithm component is called to participate in the search operations. The GA component generates a new population using the two evolutionary operators, crossover and mutation, and a copy of the current population list. The GA continues searching the solution space as long as there is improvement from one generation to another. Every time a new population is generated, GA also compares the quality of the best member of the new population with that of the overall best solution to update the original population list. Once the GA detects no improvement within its module, it will return the original population (that may have been also updated by the GA) and the best solution found during its own operations to the Tabu-SA component. The Tabu-SA utilizes the best solution (provided by GA) as the initial solution. The framework of the CSA is presented in Figure 5. In the figure, controls 1 and 5 check if the overall best solution has been improved to update the population list. Control 2 verifies the improvement within the GA component, and controls 3 and 4 test whether or not the short and long iterations are completed respectively.

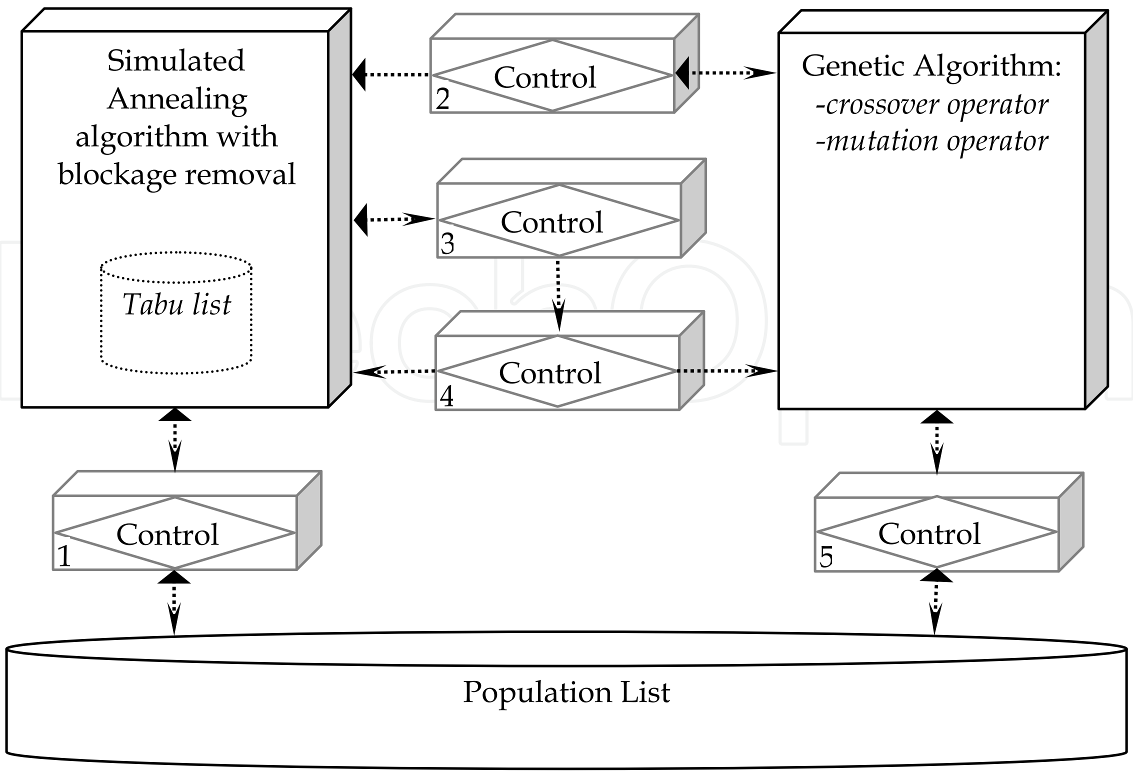


Fig. 5. Framework of the CSA

The CSA could be regarded as a variation of the SAMED algorithm proposed for the flow shop scheduling problem, SAMED-FSS (Azizi *et al.*, 2009b). However, it differs from the SAMED-FSS algorithm in several aspects. The main differences between the two algorithms have been highlighted in Table 1.

CSA(Collaborative Search Algorithm)	SAMED-FSS Algorithm
Population list is continually updated by SA and GA throughout the search	Long-term memory is updated only at the end of short iterations and is emptied at the end of long iterations
The population list is initially populated by randomly generated solutions which are gradually replaced by overall best solutions as the search progresses	Long-term memory contains the short iteration's best solutions that have been visited during the recent long iteration
Contains a complete genetic algorithm	GA component include a crossover operator
Genetic algorithm operates as long as there is improvement from one generation to another	Genetic algorithm component operates only for one generation
Once GA stops functioning, it returns the best solution found within the GA module	The new population is scanned and the first offspring whose cost is different than that of the iteration best solution might be selected.

Table 1. Differences between CSA and SAMED-FSS algorithm

4. Application of the CSA to the Permutation Flow Shop Scheduling Problem

4.1 Flow shop scheduling problem

The permutation flow shop scheduling problem studied in this paper can be described as follows. Consider a set of machines (M_1, M_2, M_3,…, M_m) and a set of jobs (J_1, J_2, J_2,…, J_n). Each job has to be processed on all m machines in the order given by the index of the machines. Each job consists of a sequence of m operations, O_{1j}, O_{2j} , O_{3j},…,O_{mj}, each of them corresponding to the processing of job j on machine i during an uninterrupted processing time period $P_{i,j}$. Each machine can only process one job at a time and it is assumed that each machine processes the jobs in the same order. The objective is to find the schedule that has the minimum makespan (the duration in which all jobs are completed).

4.1.1 Solution representation

A solution of the flow shop scheduling problem is represented by a string composed of several elements each corresponding to a job number (job based scheme). The sequence of the job numbers on the string indicates the processing order of jobs on all machines.

4.1.2 Neighbouring solution

A neighbouring solution is generated by inserting a randomly selected job in front or behind another job in the string (i.e., insertion technique).

4.2 Components of CSA

As noted earlier, the CSA includes two main components, a Tabu-SA module with blockage removal and a genetic algorithm. The Tabu-SA component of the CSA is adopted from (Azizi *et al*. (2009b). The genetic algorithm component includes a random selection mechanism, a crossover operator, and a mutation operator. In order to generate a new population, two parents are selected randomly and operated by the Precedence Preservative Crossover (PPX) (Bierwirth *et al*., 1996). To generate a new offspring by the PPX crossover operator, a template vector h of length *n* (*n* denotes the number of jobs) is filled with random elements from the 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 both parents. This procedure repeats itself until both parent strings are emptied and the offspring contains all the involved elements.

The mutation operator selects a job randomly and inserts it in front or behind another job in the offspring chromosome. The corresponding makespans of the offspring chromosome before and after the mutation operation are compared and if the mutation deteriorates the quality of the offspring, the mutation result is revoked and then the chromosome is added to the population.

4.3 Initial population and initial solution

Both the initial population and initial solution in the CSA could be generated randomly. However, for the case of flow shop scheduling we developed a constructive heuristic based on the NEH algorithm (Newaz *et al*., 1983) to generate portion (e.g., 50% of the members) of the population. According to this constructive heuristic, a solution is first generated randomly. Then, the solution is scanned from left to right and the first two jobs on the solution string is selected and added to a template chromosome. Following this, the positions of the two jobs is swapped and the partial makespans of the template chromosome before and after the swap is calculated. The makespans are compared and the configuration that generates shorter makespan is selected. Once the configuration of the two jobs is decided, the next job on the original solution (i.e., third job from left) is inserted in all possible positions on the template chromosome (e.g., before the first job, between job one and job two, and after job two). The lengths of the schedules associated with each configuration are calculated and the one with shorter makespan is selected. The above steps are repeated until all jobs on the randomly generated solution are transferred to the template chromosome.

The rest of the population members, i.e., the second portion, are generated randomly. The best member of the population is selected as the initial solution for the Tabu-SA algorithm.

5. Computational Results

The performance of the CSA is evaluated using 40 hard problems selected from two benchmark problem sets known as Taillard (1993) and Demirkol *et al*. (1998) benchmark problems. The algorithm is coded in Visual Basic and run on a Pentium 4 PC with 3.3 GHz CPU. The computational times as well as the parameters of the CSA including the size of tabu list (short iteration), long iteration (long-term memory), the initial temperature, the control parameter, and the maximum number of unacceptable moves are the same as those utilized in the SAMED-FSS algorithm (Azizi *et al.*, 2009b). The size of the population list is 30 for all benchmark problems except for TA21, TA22, TA24, and TA25 for which it is set to 20. For all benchmark problems, 50% of the members in each initial population are generated using the modified NEH algorithm described in section 4.2 and the remaining solutions are generated randomly.

The computational results for both benchmark problem sets are summarized in Table 2. The results presented in this table correspond to the best makespan over 10 runs and associated computing time (per second). According to the results presented in Table 2, the CSA found solutions with shorter makespans for 24 out of 28 Demirkol *et al.* (1998) benchmark problems (DMU01-DMU34) compared to those provided by the SAMED-FSS (Azizi *et al.*, 2009b). In comparison with the PSO (Liao *et al.*, 2007) algorithm, solutions found by the CSA for 22 benchmark problems are of better quality. The CSA also outperformed the conventional SA, standard GA, and hybrid genetic algorithm with local search (GA-LS) (Azizi *et al.*, 2009b) in all 28 test problems.

Furthermore, For all the 12 Taillard (1993) benchmark problems, the CSA found better quality solutions (including nine optimal) compared to those solutions found by the conventional SA, standard GA, hybrid genetic algorithm with local search (GA-LS), and the PSO (Liao *et al.*, 2007). Moreover, the CSA found the same (optimal) solutions as the SAMED-FSS for nine problems but with shorter computational times in the majority of the (nine) cases. In the remaining three problems, the CSA outperforms the SAMED-FSS both in terms of solution quality and computational times.

Figure 6 compares the standard deviation of the makespans obtained by five different methods for all the Demirkol *et al.* (1998) benchmark problems. The results presented in the figure clearly show that in the majority of the cases investigated in this study, the standard deviation of the makespans obtained by the CSA is significantly lower than those obtained by the other four techniques.

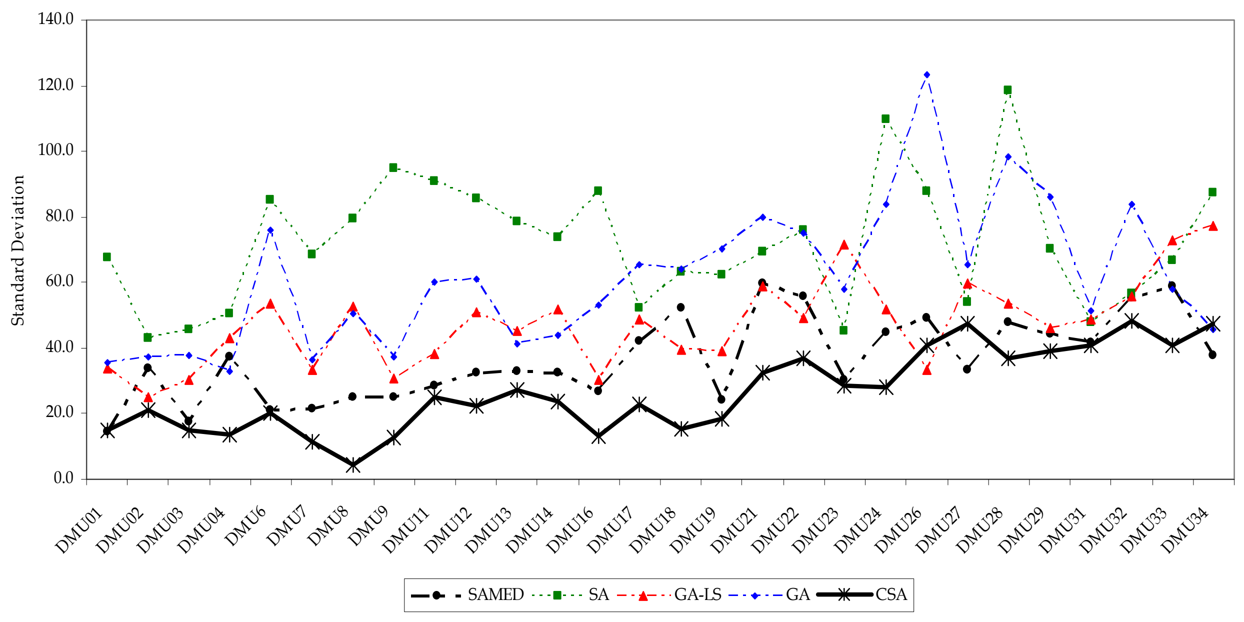


Fig. 6. Comparison of standard deviation of makespans for Demirkol *et al.*, (1998) benchmark problems

Size (n×m)[a]	Problem	Optimal	Demirkol *et al.*, 1998	SA[b]	GA[b]	GA-LS[b]	PSO: Liao *et al.*, 2007	SAMED-FSS (Azizi *et al.*, 2009b)	CSA
20 × 15	DMU01	-	4437(69)	3965(5)	3981(55)	3981(5)	3937	3899(4)	**3896(13)**
	DMU02	-	4144(0.09	3838(2)	3878(2)	3833(2)	3571	3761(13)	3755(55)
	DMU03	-	3779(58)	3579(8)	3625(1)	3572(3)	3981	3534(7)	**3532(5)**
	DMU04	-	4302(67)	4097(6)	4148(25)	4094(4)	3805	4032(51)	4032(33)
20 × 20	DMU06	-	4821(159)	4609(5)	4634(78)	4577(6)	4612	4523(40)	**4523(25)**
	DMU07	-	4779(148)	4544(12)	4561(9)	4482(10)	4570	4428(23)	**4424(47)**
	DMU08	-	4944(176)	4601(6)	4604(30)	4531(8)	4504	4523(79)	4520(55)
	DMU09	-	4886(203)	4590(6)	4574(34)	4546(8)	4538	4496(48)	**4496(10)**
30 × 15	DMU11	-	5226(149)	4612(36)	4681(74)	4682(6)	4658	4582(22)	**4574(58)**
	DMU12	-	5304(163)	4742(55)	4730(79)	4725(10)	4743	4675(61)	**4669(76)**
	DMU13	-	5079(109)	4670(13)	4771(46)	4704(8)	4824	4569(99)	**4568(63)**
	DMU14	-	5605(0.17	4894(40)	5014(38)	4949(12)	4928	4836(54)	**4836(52)**
30 × 20	DMU16	-	6183(0.24	5431(48)	5494(59)	5494(19)	5782	5384(76)	**5372(87)**
	DMU17	-	6037(471)	5815(4)	5854(53)	5794(17)	5485	5718(75)	5700(28)
	DMU18	-	6241(394)	5815(4)	5856(60)	5815(22)	5486	5726(55)	5767(43)
	DMU19	-	6095(320)	5546(57)	5654(34)	5537(20)	5848	5479(68)	**5469(85)**
40 × 15	DMU21	-	6986(156)	6089(69)	6229(83)	6073(17)	6012	6026(119)	**5965(107)**
	DMU22	-	6351(224)	5797(95)	5931(101)	5829(14)	6080	5717(100)	**5702(117)**
	DMU23	-	6506(289)	5944(86)	6105(60)	5997(15)	6173	5904(40)	**5895(108)**
	DMU24	-	6845(186)	5967(33)	6081(102)	6039(13)	5855	5928(55)	5928(36)
40 × 20	DMU26	-	7154(615)	6711(31)	6771(98)	6763(34)	6730	6556(176)	**6518(167)**
	DMU27	-	7528(645)	6833(140)	6972(161)	6824(35)	6723	6704(178)	**6686(90)**
	DMU28	-	7469(674)	6965(173)	7033(160)	6962(29)	6973	6860(180)	**6842(144)**
	DMU29	-	7608(682)	6834(131)	7000(133)	6935(23)	6950	6792(166)	**6749(76)**
50 × 15	DMU31	-	7673(313)	6747(92)	6952(76)	6852(14)	6725	6747(80)	**6677(168)**
	DMU32	-	7679(299)	6740(172)	6844(118)	6761(28)	7143	6669(89)	**6614(49)**
	DMU33	-	7416(284)	6733(134)	6893(144)	6748(31)	6864	6656(138)	**6596(117)**
	DMU34	-	7548(307)	6945(149	7125(159)	6972(28)	7070	6878(101)	**6839(152)**
20 × 10	Ta11	1582	-	1593(110)	1622(3)	1586(4)	1604	1582(41)	1582(47)
	Ta12	1659	-	1676(33)	1710(4)	1692(1)	1685	1659(67)	1659(46)
	Ta13	1496	-	1524(38)	1540(1)	1515(1)	1520	1496(156)	1496(31)
	Ta14	1377	-	1384(118)	1413(10)	1396(2)	1402	1377(70)	1377(17)
20 × 20	Ta21	2297	-	2320(38)	2325(2)	2315(7)	2319	2297(12)	2297(7)
	Ta22	2099	-	2125(32)	2120(23)	2127(9)	2132	2099(18)	2099(28)
	Ta24	2223	-	2254(7)	2264(5)	2246(29)	2243	2223(17)	2223(14)
	Ta25	2291	-	2315(83)	2325(6)	2309(46)	2315	2291(43)	2291(12)
50 × 10	Ta41	2991	-	3059(208)	3116(71)	3111(21)	3080	3035(148)	**3030(146)**
	Ta42	2867	-	2921(197)	3006(77)	2975(9)	2974	2911(97)	2911(117)
	Ta44	3063	-	3071(298)	3194(71)	3121(17)	3103	3066(217)	**3063(201)**
	Ta48	3037	-	3048(230)	3121(88)	3108(10)	3085	3044(116)	**3042(164)**

Table 2. Comparison of computational results

a) n×m = n jobs and m machines, b)Conventional SA, Standard GA, and GA with Local Search (Azizi *et al.*, 2009b); **bold**= best solution.

6. Conclusion

Various scheduling problems that occur in manufacturing industries have been investigated in the literature. They are inherently complex and often referred to as combinatorial Non-Polynomial (NP) hard problems. These problems are very difficult to solve using existing heuristics or conventional techniques. This chapter presents a generic framework of a collaborative search algorithm to solve scheduling problems. The proposed framework contains two independent search modules that exchange information while consecutively run to solve a problem. Based on the proposed framework a search algorithm tailored for the flow shop scheduling is presented. The computational results for the two challenging classical problem sets clearly indicate the superior performance of the proposed method over several conventional techniques including a simulated annealing, a genetic algorithm and a hybrid genetic algorithm. The CSA results also compare favourably with those of the two newly developed algorithms, PSO (Liao *et al.*, 2007) and SAMED-FSS (Azizi *et al.*, 2009b).

7. References

Aiex, R.M.; Binato, S.; & Resende, M.G.C. (2003). Parallel GRASP with path relinking for job shop scheduling. *Parallel Computing*, 29, 393-430.

Aydin, M.E. & Fogarty, T.C. (2004). A Distributed Evolutionary Simulated Annealing Algorithm for Combinatorial Optimisation Problems. *Journal of Heuristics*, 10, 269–292.

Azizi, N. & Zolfaghari, S. (2004). Adaptive temperature control for simulated annealing: a comparative study. *Computers and Operations Research*, 31, 2439-2451.

Azizi, N.; Zolfaghari, S.; & Liang, M. (2009a). Hybrid simulated annealing with Memory: An evolution-based diversification approach, *International Journal of Production Research (IJPR),* accepted.

Azizi, N.; Zolfaghari, S.; & Liang, M. (2009b). Hybrid Simulated Annealing in Flow-shop Scheduling: A Diversification and Intensification Approach, *International Journal of Industrial and Systems Engineering (IJISE)*, 4(3), 326-348.

Bierwirth, C.; Mattfeld, D.C.; & Kopfer, H. (1996). On Permutation Representations for Scheduling Problems, *proceeding of 4th International Conference on Parallel Problem Solving from Nature*, Lecture notes in computer science, Springer-Verlag, pp.310–318.

Demirkol, E.; Mehta, S.; & Uzsoy, R. (1998). Benchmarks for shop scheduling problems. *European Journal of Operational Research*, 109 (1), 137-141.

Dorigo, M. & Gambardella, L.M. (1997). Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transaction on Evolutionary Computation*, 53-66.

El-Bouri, A.; Azizi, N.; & Zolfaghari, S. (2007). A comparative study of new metaheuristics based on simulated annealing and adaptive memory programming. *European Journal of Operations Research*, 2007, 177, 1894-1910.

Feo, T. & Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 16, 109-133.

Gonçalves, J.F.; Mendes, J.J.M.; & Resende, M.G.C. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 2005, 167, 77-95.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI.

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. *Proceeding of. IEEE International Conference on Neural Network*, 1942–1948.

Kirkpatrick, S.; Gelatt, C.D. Jr. & Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, 220, 671-680.

Kolonko, M. (1999). Some new results on simulated annealing applied to the job shop scheduling problem. *European Journal of Operations Research*, 113, 123-136.

Liao, C.J.; Tseng, C.T.; & Luarn, P. (2007). A discrete version of particle swarm optimization for flow shop scheduling problems. *Computers & Operations Research*, 34(10), 3099-3111.

Nawaz, M.; Enscore, E. E. & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *OMEGA*, 11(1), 91-95.

Osman, I.H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41(4), 421-451.

Reeves, C. R. (1995). A genetic algorithm for flow shop sequencing. *Computers & Operations Research*, 22(1), 5-13.

Ruiz, R. & Maroto, C. (2005). A comprehensive review and evaluation of permutation Flow shop heuristics. *European Journal of Operational Research*, 165(2), 479-494.

Taillard, E. (1993). Benchmark for basic scheduling problems. *European Journal of Operational Research*, 64(2), 278-285.

Taillard, E.D.; Gambardella, L.M.; Gendreau, M. & Potvin, J.Y. (2001). Adaptive memory programming: A unified view of metaheuristics. *European Journal of Operational Research*, 135, 1-16.

Wang, L. & Zheng, D.Z. (2003). An effective hybrid heuristic for flow shop scheduling. *International Journal of Advanced Manufacturing Technology*, 21(1), 38-44.

Zolfaghari, S. & Liang, M. (1999). Jointly solving the group scheduling and machining speed selection problems: A hybrid tabu search and simulated annealing approach. *International Journal of Production Research*, 37(10), 2377-2397.