Combining the large-step optimization with tabu-search: application to the job-shop scheduling problem

Combining the Large-Step Optimization with Tabu-Search: Application to The Job-Shop Scheduling Problem

Helena Ramalhinho Lourenço
Departamento de Estatística e Investigação Operacional
Faculdade de Ciências
Universidade de Lisboa
Campo Grande C/2
1700 Lisboa, Portugal
e-mail: helena@gist.deio.fc.ul.pt

Michiel Zwijnenburg
Department of Econometrics
Faculty of Economy
Erasmus University
Rotterdam, The Netherlands

Abstract:

We apply the combined technique of tabu-search and large-step optimization to the job-shop scheduling problem. The job-shop scheduling problem can be defined as follows: given a set of machines and a set of jobs, the objective is to construct a schedule which minimizes the time necessary to complete all the jobs. We also present some diversification strategies for the tabu-search methods and relate them with the large-step optimization method. Relevant computational results and respective conclusions are also presented.

Key Words: Local Search, Tabu Search, Large-Step Optimization, Job-Shop Scheduling.

1. Introduction

We apply the combined technique of tabu-search and large-step optimization to the job-shop scheduling problem. In the job-shop scheduling problem we are given a set of m machines $\{M_1,...,M_m\}$ and a set of n jobs $\{J_1,...,J_n\}$. Each job J_j, j=1,...,n, consists of a sequence of m_j operations O1j,..., $O_{mj,j}$, where O_{ij} is an operation of job J_j to be processed on machine μ_{ij} for a given uninterrupted processing time p_{ij}, where $\mu_{ij\_}$ μ_{i+1j}, for i=1,...,m_j, j=1,...,n.

The operations of each job must be processed in the given sequence. Each machine M_i, i=1,...,m, can process at most one operation at a time, and at most one operation of each job J_j, j=1,...,n, can be processed at a time. Let C_{ij} be the completion time of operation O_{ij}. The objective is to get a schedule that minimizes the maximum completion time $C_{max}=max_{i,j}$ C_{ij}. A schedule is an allocation of a single time interval for each operation. A more useful representation is provided by the disjunctive graph model that can be represented as follows: G=(O,A,E), where O is the vertex set corresponding to the operation set, A is the arc set corresponding to the job precedence constraints and E is the edge set corresponding to the machine capacity constraints.

In the disjunctive graph, the basic scheduling decision corresponds to orienting each edge in one direction or the other. A schedule is obtained by orienting all edges. The orientation is feasible if the resulting directed graph is acyclic, and its length is equal to the weight of a maximum weight path from s to t in this acyclic graph.

The job-shop scheduling problem is *NP*-hard in the strong sense. Furthermore, most of the special cases of this problem are *NP*-hard or strongly *NP*-hard.

In Section 2, we review local optimization methods, as local improvement and simulated annealing, and a two-phase optimization method, known as large-step optimization, which has recently been introduced for the traveling salesman problem. The first phase of this new method consists of a large optimized transition in the current solution, while the second phase is basically a local search method.

The main advantage of the large-step optimization methods is the use of optimization and tailored procedures in combination with local search methods.

So far, the methods used in this second phase, also called small-steps phase, are the local improvement and the simulated annealing methods. In this work we combine the large-step optimization with a tabu-search approach and apply it to the job-shop scheduling problem, because both methods separately applied to the problem gave good results. We present this combined method and related computational results in Section 3.

In Section 4, we present some diversification strategies for the tabu-search methods and relate them with the large-step optimization method. Relevant computational results are also presented. In Section 5, we present some conclusions and possible future research ideas.

2. Local and large-step optimization methods

Local improvement methods are iterative methods where initially a feasible solution of the problem in question is obtained, and at each step we make one local modification, or transition, of a prespecified type in the current solution. If an improvement in the cost function is obtained, then we accept the new solution as the current one, and, otherwise, we reject it. The algorithm terminates when we cannot improve the current solution by performing one transition, i.e., when we have a local optimal solution.

Consider now the job-shop scheduling problem. A solution corresponds to a schedule and the cost function is the maximum completion time of the schedule. The method used to obtain the initial schedule is the priority rule with priority function most work remaining. An extensive study considering different methods to obtain initial schedule has been done by Lourenço (1995).

Simulated annealing accepts solutions with increased value for the cost function, called uphill moves, with small and decreasing probability, in an attempt to get away from a local optimal solution and keep exploring the region of the feasible solutions. The probability is controlled by a parameter known as the temperature, which is gradually reduced from a high value, at which most uphill moves are accepted, to a low one at which few, if any, such moves are accepted.

In our implementation of simulated annealing, we use the same method to generate the initial schedule as for the local improvement method, and the neighborhood structure of van Laarhoven, Aarts and Lenstra (1992) (neighborhood VLA&L), but we use a different cooling schedule. Our cooling schedule is very similar to the geometric cooling schedule presented in Johnson et al.(1989). Later on, we also consider a version of the simulated annealing method using the neighborhood structure developed by Dell´Amico & Trubian (1993), (neighborhood D&T).

Martin, Otto and Felten (1992) introduced a large-step optimization method for the traveling salesman problem, which was applied to the job-shop scheduling problem by Lourenço (1993). The large-step optimization method consists of three main routines: a large-step phase, a small-steps phase and an accept/reject test, which all three are consecutively executed for a certain number of large-step iterations.

Local improvement methods proceed downhill for a while, making good progress, but they tend to get trapped in local optimal solutions, usually far away from the global optimal solution. Simulated annealing methods try to improve this by accepting uphill moves depending on a decreasing probability controlled by the temperature parameter. But, at small temperatures, they also tend to get stuck in valleys of the cost function. Large-step optimization methods allow to leave these valleys even at small temperatures, and only then a local search optimization method is applied, returning to a local optimal solution. At this point, an accept/reject test is performed.

The three main routines of a large-step optimization method are: the method to perform a large step, the one for the small steps and the accept/reject test (see Figure 1).

An important part of the large-step optimization methods is the large-step phase. The large step phase should be constructed so that valleys are easily climbed over, and they should be specially tailored to the problem in consideration. As mentioned, this procedure should make a large modification in the current solution to drive the search to a new region and, at the same time, to perform some kind of optimization to obtain a solution not too far away from a local (or global) optimal schedule. These two factors of the large-step method are the main differences from usual local search methods.

```
1. Get an initial schedule $s_0$.
2. Perform the following loop *numiter* times:
    2.1. Large step phase: change $s_i$ to $s´_i$ by performing
         a big perturbation.
    2.2. Small steps phase: run a local search
    optimization method with initial
    solution $s´_i$, and obtain $s''_i$.
    2.3. Perform an accept/reject test comparing $s_i$ with
         $s''_i$, if $s''_i$ is accepted, let $s_{i+1}=s''_i$.
3. Return the best solution found $s_{best}$.
```

Figure 1: Large-step optimization method.

So far, the methods that are usually associated with the small steps phase are local optimization algorithms like the local improvement method or the simulated annealing method. In the next section, we will consider the tabu-search approach as the small-steps procedure.

The accept/reject test can accept only downhill moves. This looks like the local improvement methods presented before, but where only local optimal solutions (with respect to the small steps) are considered. On the other hand, the accept/reject test can accept all moves or it can also look like the test done in a simulated annealing method, i.e. all downhill moves are accepted and a uphill move is accepted with a small and decreasing probability.

Different applications of the large-step optimization methods have appeared in the literature. Martin, Otto and Felten (1992) developed a large-step optimization methods for the traveling salesman problem, where they used a 4-opt move as the large step phase, and, as the small steps, a local improvement method based on 3-opt moves. Johnson (1990) also applies a large-step local optimization method, that he called iterated Lin-Kernighan, to the traveling salesman problem, using a 4-opt move as the large-step and the Lin and Kernighan heuristic for the small steps. Lucks (1992) applied large-step optimization to the graph partitioning problem, where the large step is done by interchanging a randomly selected set of vertices of a given size, and, as small steps, the Kernighan and Lin approximation method and a simulated annealing approach were used. He concluded that for random graphs, large-step optimization outperformed all other tested algorithms, while for geometric graphs, the large-step method in some cases was outperformed. Brucker, Hurink & Werner (1993),(1994), presented a version of the large-step methods, designated as improving local search heuristics, for some scheduling problems.

The large step optimization is different from random restarts methods, Feo and Resende (1994), in which randomly constructing an initial schedule followed by a local optimization method is repeated for a number of iterations, returning the best solution found. As suggested by Johnson (1993), large-step optimization methods can be viewed as restart methods. But instead of starting with a different initial random solution at each iteration, in the large step a solution s´ is obtained from a solution s, by performing a sufficient large optimized perturbation. In this way, all the achievements from previous runs are not totally lost in the next runs. After every large step a local optimization method is applied as a small steps procedure which leads the search process to a good local optimal solution. Also, these methods have the advantage of combining the ideas associated with local search methods and tailored heuristics.

Next, we apply the large-step optimization methods to the job-shop scheduling problem. For the small steps phase, we have considered the following methods: the local improvement method and the simulated annealing method. Lourenço (1995) proposed four methods to perform the large step procedure. The first three methods are based on choosing one or two machines and reordering the operations to be processed on these machines, using some prespecified method, and the last one is based on reversing the order of processing of several operations. We use one of the most successful which can be briefly described as follows: reschedules iteratively two randomly chosen machines to optimality. First choose two random machines and ignore the order of the operations to be processed in these machines. Next, for one of the two machines, determine the release and delivery times of the operations to be processed by the respective machine, given the scheduled operations on the other machines. Solve the one-machine schedule associated with this machine using Carlier's algorithm, Carlier (1982). Then repeat this process to reschedule the second machine. For more information, see Lourenço (1995).

Next, we present computational results obtained from the application of these methods to several instances of the problem (see Tables 1 to 4). The methods are tested on set of known instances. Since the amount of testing is extensive, in preliminary tests we use just a subset of the instance. All programs were written in C. For instances of size 10 jobs, 10 machines, and smaller, the programs run in a Shine 486DX with a clock of 66 Mhz, and for the bigger instance the programs ran on a SPARC SUN 4M.

For the job-shop scheduling problem, Lourenço (1995) concluded that the local improvement method is clearly inferior to the simulated annealing, even when several trials are performed. Also the large-step method combined with the local improvement gave no better results than when we combined the large-step with the simulated annealing. Therefore we do not consider the local improvement in our computational experience. The large-step optimization methods outperformed the simulated annealing method, but the difference between these two became closer when the same amount of time was allowed for both. In many cases, the large-step optimization methods found an optimal schedule and in others the distance from the optimum or lower bound was small.

Given the results obtained by previous applications of the large-step optimization methods to combinatorial optimization methods, including the job-shop scheduling problem, and given the success of tabu-search to the same problem, it looks promising to apply the large-step optimization method using in the small steps phase a tabu-search method.

3. Combined techniques of large-step optimization and tabu-search

The tabu-search, in contrary of local optimization methods presented previously, makes use of historical information. The historical information is kept in three kinds of memory functions: the short-term, the intermediate and long term memory function. The last two will be considered in next section. Short term memory function has the task to memorize certain attributes of the search process of the recent past and it is incorporated in the search via one or more tabu list. For more information on tabu-search methods see, for example, Glover (1989) and Glover et al.(1993).

The application of tabu search to the job-shop scheduling problem by Dell'Amico and Trubian (1992) gave very good results, taking a small amount of time. Note that a big percentage of the running time for a large-step optimization method is spent by the simulated annealing method. Therefore, an improvement for large-step optimization methods should be achieved by using small-step methods that give good local optimal solutions and run in a shorter amount of time than the simulated annealing. By the above observations the tabu search methods looks like a good candidate to be used instead of the local improvement method, that are fast but gives poor local optimal solutions, and instead of simulated annealing that gives good local optimal solutions, but at very high computational cost.

The basis for our implementation of the tabu-search is derived from Dell'Amico and Trubian (1993), but there are some differences. Dell'Amico and Trubian used a bi-directional method to obtain an initial schedule, while our tabu-search, like in the local improvement method and simulated annealing approach, starts by constructing a initial schedule according to the priority rule most remaining work. In our implementation we did not apply their intensification strategy which let the process return to the best solution found so far in the process, if during a certain number of iterations the best solution did not improve. The reason for this is because we plan to use the tabu-search in the small-steps phase of the large-step optimization method. Also, different in our implementation is the use of exact methods for calculating the longest path in a graph belonging to a feasible solution and for checking if a graph contains a cycle. Similar to Dell'Amico and

Trubian tabu-search are the neighborhood structure, the tabu-list structure and the rules which define the length of the tabulist.

The parameters in all programs are set in such a way that for a certain instance, they use the same amount of computer time as our implementation of the Dell'Amico and Trubian tabu-search method needed for 12000 iterations for the same instance. For example, for instance MT10 this took more and less 1550 seconds on a Shine 486DX with clock 66 Mhz.

Table 1: Best results for the Local Optimization Methods with VLA&L neighborhood structure.

Neighborhood: VLA&L		Size	SA	TS	LSSA	LSTS
Instance	Opt	n*m	best	best	best	best
ABZ5	1234	10*10	1239	1236	1244	1238
CAR5	7702	10*6	7822	7732	8305	7702
LA04	590	10*5	590	590	590	590
LA21	1046	15*10	1058	1056	1069	1053
MT10	930	10*10	949	930	956	940
ORB3	1005	10*10	1040	1020	1051	1005
ORB4	1005	10*10	1026	1019	1026	1011

Table 2: Average results for the Local Optimization Methods with VLA&L neighborhood structure.

Neighborhood: VLA&L		Size	SA	TS	LSSA	LSTS
Instance	Opt	n*m	av.	av.	av.	av.
ABZ5	1234	10*10	1247	1237	1245	1239
CAR5	7702	10*6	8019	7943	8476	7924
LA04	590	10*5	591	590	591	590
LA21	1046	15*10	1080	1060	1077	1060
MT10	930	10*10	960	945	973	942
ORB3	1005	10*10	1060	1027	1064	1015
ORB4	1005	10*10	1035	1022	1030	1013

The results are presented in Table 1, 2, 3 and 4. We can conclude that the large-step optimization combined with the tabu-search outperforms the simulated annealing and the large-step optimization using simulated annealing. The average value out of 5 runs for the large-step optimization with tabu-search is always smaller than the average results for the large-step optimization with simulated annealing. In the beginning of the search in the simulated annealing approach there is no fast improve as in the tabu-search. Therefore the simulated annealing wastes time in the beginning of the process, meanwhile this initial period is effectively used by tabu-search. Even the tabu-search alone outperforms the large-step optimization method with simulated annealing small-steps procedure but it is slightly inferior to the large-step optimization with tabu-search. For example, for the famous instance of Muth and Thompson with 10 jobs and 10 machines (MT10) and considering the D&T neighborhood, the best value obtained by the large-step optimization method with tabu search (LSTS) was the optimal value of 930 and the average was 939 meanwhile if the simulated annealing (LSSA) was used as the small-steps procedure the respective values were 951 and 963. Using our implementation of the tabu-search (TS) alone we obtained the following values, 940 was the best obtained and 946 the average. For the instance propose by Lawrence (LA21) the best value obtained by LSTS was 1047, and the average was 1055. While for the LSSA the respective values were 1078 and 1096. Applying the TS method alone we obtained the following the values 1059 and 1065.

Table 3: Best results for the Local Optimization Methods with D&T neighborhood structure.

Neighborhood: D&T		Size	SA	TS	LSSA	LSTS
Instance	Opt/LB	n*m	best	best	best	best
ABZ5	1234	10*10	1245	1238	1238	1238
CAR5	7702	10*6	7835	7725	7725	7720
LA04	590	10*5	597	590	590	590
LA21	1046	15*10	1090	1059	1078	1047
MT10	930	10*10	944	940	951	930
ORB3	1005	10*10	1048	1023	1069	1024
ORB4	1005	10*10	1035	1013	1021	1013

Table 4: Average results for the Local Optimization Methods with D&T neighborhood structure.

Neighborhood: D&T		Size	SA	TS	LSSA	LSTS
Instance	Opt/LB	n*m	av.	av.	av.	av.
ABZ5	1234	10*10	1257	1240	1247	1239
CAR5	7702	10*6	7905	7790	8108	7940
LA04	590	10*5	605	591	598	591
LA21	1046	15*10	1100	1065	1096	1055
MT10	930	10*10	964	946	963	939
ORB3	1005	10*10	1073	1034	1077	1030
ORB4	1005	10*10	1056	1019	1027	1022

We have also compared several large-step optimization methods, which differ from each other by the number of the large-step iterations. The combination of large/small steps number of iterations are set in a special proportion to each other such that the complete run of the algorithms takes as much time as one run of the previous large-step optimization method with tabu-search, 12000 total iterations. In general the best results in terms of the best and the average value were obtained by large step optimization with the number of large step iterations between 5 and 20, with a little drop near 10 large-step iterations.

Viewing the results for both tabu-search with different neighborhoods, it seems that the simple neighborhood structure (VLA&L) with a large number of iterations can compete with a more complicated neighborhood structure (D&T), which need more time to pick the best move out of all possible ones, and therefore may be executed for fewer iterations. But more tests in this issue is need to be able to make stronger conclusions.

4. Diversification strategies

Long term memory is the basis for the diversifying strategy which has to lead the process to new regions of the solution space. The objective of a diversification strategy is to produce a new starting point, wherefore the local search can continue.

In this section we pay attention to a number of diversification strategies, which are closely related to the large-step phase. Both a diversification strategy as well as a large-step phase apply a big change to the current solution, which provides them the possibility to get the search process out of a local optima valley. The difference between the large-step procedure and the diversification strategies that we are proposing is that this latter ones make use of a long term memory, where, large-step optimization can apply a relevant big change to a given solution without the use of information directly from the past, but using optimization techniques.

In the approach to obtain a new starting point, two kind of diversifying strategies can be distinguished, strategies using a restart method and strategies using a method which lead the search to new regions by an iterative process. These last kinds of diversification methods usually provide advantages over restart methods, Glover (1993). For a number of iterations the random restart methods construct a random initial schedule, followed by the application of a local optimization procedure, returning the best solution found. The only difference with a more systematic restart method is that the initial solution is not randomly obtained, but constructed with information from previous periods in the search process. The storage of this information can be done with frequency counts. The idea is to count the number of times that moves are applied during the non-diversification phase of the search process, while during the diversification phase the moves are penalized according to their frequency counts, with the objective to oblige the process to search for moves not so often applied, resulting in never visited solutions.

Next, we present two diversification methods for the job-shop scheduling problem using frequency counts. The first method is a restart method which begins every diversification iteration by constructing a new starting schedule using frequency counts obtained in the preceding steps of the method, that we will call small steps in analogy with the large-step optimization methods. Every time the method finds a better solution in the small steps phase, best solution and best value are updated and the frequency memorizes the predecessor and successor of every job on every machine in the new best schedule. After these period of small-steps is finished, the frequency counts for every job how many times a job was the predecessor and the successor of any another job at any machine in the improving schedules. We obtain a diversified schedule by constructing a schedule according to a priority rule, which can be overruled if it wants to schedule an operation after another operation resulting in an already existing neighbor combination in one of the memorized improving schedules. In this way, we try to obtain a schedule wherein the predecessor and successor of every job are different from the predecessors and successors of the same job in the same machine in the memorized improving schedules.

The second diversification we have implemented is an iterative method, using frequency counts, which leads on a path to new regions of the solution space. The frequency counts are obtained in the previous iterations by storing the number of times a certain arc has been reversed. In our implementation we memorized only the critical arcs, involved in a move. This is obvious for the VLA&L neighborhood. For the D&T neighborhood structure, counting the exact number of reversals would be very time consuming if we consider all single reverses. Therefore, we decided to count only the number of time the critical arcs are involved in a move. The diversification method consist of running the same tabu-search method, but now the moves are penalized to the critical arcs involved in the following way (see Laguna (1992)):

compare-length(s´)=length(s´)+ penalty*frequency-of-critical-arc,

where s´ is the neighbor of s. If a critical arc of s is involved in the applied move, the number of times this arc was picked in the preceding steps is stored in the frequency-of-critical-arc. The penalty is the value which a move is punished per applied move involving the arc. In this way moves which were frequently applied in the preceding iterations are punished in the diversification phase to force the method to apply new moves, with the objective to drive the search process to new regions. The value of the variable penalty and the number of iterations in the diversification phase will be determined via trial and error method.

We obtained results with this method for four combinations of number of diversification steps/small steps iterations (see Tables 5, 6, 7 and 8). These combinations are similar to the ones used in testing the large-step optimization methods with tabu-search. With some exceptions the incorporation of our restart diversification strategy does not contribute to a better performance of the tabu-search method. We have observed that after the diversification iterations, a resulting schedule has a high length. In general, the best values, as well the average ones, are getting worse when the number of diversification steps increases. Therefore we can conclude that our restart diversification strategy changes the schedule too drastically. After the application of the diversification method, the tabu-search method needs a lot of iterations to get near to good solutions. For the method with 5 diversification step iterations, the good solutions are already difficult to reach. In general, the results are not better than the results of tabu-search without diversification.

Table 5: Best results obtained by the tabu-search method, large-step method with tabu-search and tabu-search with restart diversification strategy

Tabu-Search with restart diversification strategy						
best	Method		Number of diversification step iterations			
Instance	TS	LSTS	5	10	15	20
ABZ5	1238	1238	1238	1238	1238	1238
CAR5	7725	7720	7821	7843	7787	8039
LA04	590	590	590	590	590	590
LA21	1059	1047	1053	1068	1079	1082
MT10	940	930	940	940	940	940
ORB3	1023	1024	1020	1026	1035	1035
ORB4	1013	1013	1011	1011	1013	1017

Consider now the second diversification strategy, designated by iterative diversification strategy. When we compare the results obtained by this method (TSID) with tabu-search without diversification (TS) and large-step optimization with tabu-search (LSTS), we can conclude that on average this method can compete with LSTS and performs better than TS. As for example, for MT10 the best and average value obtained by TSID were 936 and 938 (penalty=10) or 930 and 941 (penalty=5), meanwhile for TS was 940 and 946, and for LSTS were 930 and 939. For LA21, the results were, for TSID and LSTS were 1047 and 1055, respectively.

Table 6: Average results obtained by the tabu-search method, large-step method with tabu-search and tabu-search with restart diversification strategy

Tabu-Search with restart diversification strategy						
average	Method		Number of diversification step iterations			
Instance	TS	LSTS	5	10	15	20
ABZ5	1240	1239	1243	1242	1242	1240
CAR5	7790	7940	7997	8067	8022	8180
LA04	591	591	591	590	590	590
LA21	1065	1055	1065	1073	1085	1088
MT10	946	939	952	950	954	956
ORB3	1034	1030	1040	1049	1055	1053
ORB4	1019	1022	1017	1015	1017	1024

Table 7: Best results obtained by the tabu-search method, large-step method with tabu-search and tabu-search with iterative diversification strategy

Tabu-Search with iterative diversification strategy					
best	Method		Penalty		
Instance	TS	LSTS	10	2	5
ABZ5	1238	1238	1238	1236	1236
CAR5	7725	7720	7731	7702	7787
LA04	590	590	590	590	590
LA21	1059	1047	1047	1059	1060
MT10	940	930	936	937	930
ORB3	1023	1024	1023	1023	1020
ORB4	1013	1013	1013	1013	1013

Table 8: Average results obtained by the tabu-search method, large-step method with tabu-search and tabu-search with iterative diversification strategy

Tabu-Search with iterative diversification strategy					
average	Method		Penalty		
Instance	TS	LSTS	10	2	5
ABZ5	1240	1239	1239	1240	1238
CAR5	7790	7940	7935	7923	8163
LA04	591	591	591	591	591
LA21	1065	1055	1055	1073	1066
MT10	946	939	938	939	941
ORB3	1034	1030	1027	1027	1025
ORB4	1019	1022	1017	1017	1017

The reason that the iterative diversification strategy performs better than the restart one is that, the iterative diversification with the chosen parameters changes the schedule enough to get out of an eventually bad valley, but not so radical that the resulting schedule has high length. The same happen when we apply the large-step procedure in the large-step optimization methods. In the iterative diversification strategy a kind of "punished optimization" is done, and with the right parameters, it does not take a lot of small-steps iterations to get near a good solution again, contrary to what happens in the restart diversification strategy.

5. Conclusions

The idea of combining large-step optimization method with a tabu-search approach seems interesting to solve combinatorial optimization problem, due the good results that we obtain from the application to the job-shop scheduling problem. Therefore, it will be interesting to develop similar methods to solve other problems and test the efficiency of these combined optimization methods. The combination of optimization and tailored methods with local search methods may lead to an improvement in the use of these last methods in solving problem. In our opinion, it is a very interesting area to develop.

Another issue that it will be relevant to explore is the connection between the large-step procedure in the large-step optimization methods and the diversification strategies, briefly mentioned in this work. The relation with the intensification strategies was not considered in the work, but we are planning to do it in the near future.

6. References:

P. Brucker, J. Hurink and F. Werner, Improving Neighborhoods for Local Search Heuristics, Working paper, Osnabrücker Schriften zur Mathematik, Universität Osnabrück (1993).

P. Brucker, J. Hurink and F. Werner, Improving Local Search Heuristics for Some Scheduling Problems, Working Paper, Osnabrücker Schriften zur Mathematik, Universität Osnabrück (1994).

J. Carlier, The One-Machine Sequencing Problem, *European Journal of Operational Research*, 11 (1982) p. 42.

M. Dell'Amico and M. Trubian, Applying Tabu-Search to the Job-Shop Scheduling Problem, *Annals of Operations Research*, 41 (1993) p. 231.

T.A. Feo and M.G.C. Resende, Greedy Adaptive Search Procedures, to appear in *Annals of Operations Research*.

F. Glover, Tabu Search - Part I, *ORSA Journal on Computing*, 1(3) (1989) p. 190.

F. Glover, "Tabu Thresholding: Improved Search in Nonmonotonic Trajectories", to appear in *ORSA Journal on Computing*.

F. Glover, M. Laguna, T. Taillard and D. de Werra, Tabu Search, *Annals of Operations Research*, 41 (1993).

D.S. Johnson, Local Optimization and the Traveling Salesman Problem, in: *Proceedings of the 17th Annual Colloquim on Automata, Languages and Programming* (Springer-Verlag, 1990) p.446.

D.S. Johnson, Random starts for local optimization, DIMACS Workshop on Randomized Algorithms for Combinatorial Optimization (1993).

D.S Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon, Optimization by Simulated Annealing: an experimental evaluation; part I, graph partitioning, *Operations Research*, 39(3) (1989) p.865.

M. Laguna, A guide to implementing Tabu Search, *Investigacion Operativa*, (1994).

H.R. Lourenço, Job-Shop Scheduling: Computational Study of Local Search and Large-Step Optimization Methods, *European Journal of Operational Research*, 83 (1995) p.347.

V.B.F. Lucks, Large-Step Local Improvement Optimization for the Graph Partitioning Problem, Master's Dissertation, Cornell University, Ithaca, NY, USA (1992).

O. Martin, S.W. Otto and E.W. Felten, Large-Step Markov Chains for the TSP Incorporating Local Search Heuristics, *Operations Research Letters*, 11 (1992) p. 219.

P.J.M. van Laarhoven, E.H.L. Aarts and J.K. Lenstra, Job Shop Scheduling by Simulated Annealing, *Operations Research*, 40(1) (1992) p. 113.