Job shop scheduling by local search

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computing Science

Memorandum COSOR 92-29

Job Shop Scheduling by Local Search

E.H.L. Aarts
P.J.M. van Laarhoven
J.K. Lenstra
N.L.J. Ulder

Eindhoven, July 1992
The Netherlands

Eindhoven University of Technology
Department of Mathematics and Computing Science
Probability theory, statistics, operations research and systems theory
P.O. Box 513
5600 MB Eindhoven - The Netherlands

Secretariate: Dommelbuilding 0.03
Telephone: 040-47 3130

ISSN 0926 4493

Job Shop Scheduling by Local Search

E.H.L. Aarts[1,2], P.J.M. van Laarhoven[3], J.K. Lenstra[2,4] and N.L.J. Ulder[5]

[1] Philips Research Laboratories, P.O. Box 80000, NL-5600 JA Eindhoven, The Netherlands
[2] Eindhoven University of Technology, Box 513, NL-5600 MB Eindhoven, The Netherlands
[3] McKinsey & Company, Amstel 344, NL-1017 AS Amsterdam, The Netherlands
[4] CWI, P.O. Box 4079, NL-1009 AB Amsterdam, The Netherlands
[5] Océ-Nederland B.V., P.O. Box 101, NL-5900 MA Venlo, The Netherlands

Abstract

We present a computational performance analysis of local search algorithms for job shop scheduling. The algorithms under investigation are iterative improvement, simulated annealing, threshold accepting and genetic local search. Our study shows that simulated annealing performs best in the sense that it finds better solutions than the other algorithms within the same amount of running time. Compared to more tailored algorithms, simulated annealing still finds the best results but only under the assumption that running time is of no concern.

Key words: Local search, iterative improvement, threshold accepting, genetic algorithms, simulated annealing, job shop scheduling.

1 Introduction

Combinatorial optimization problems arise in such diverse areas as computer and VLSI design, facilities layout, production scheduling, and distribution planning. Many of these problems have been proved NP-hard [Garey & Johnson, 1979], and it is consequently believed that they cannot be solved in polynomial time. In practice this means that solving large instances of such problems to optimality requires impracticable running times. To avoid this, one often resorts to approximation algorithms that try to find near-optimal solutions within acceptable running times.

There is a second justification for the use of approximation algorithms. Many practical problems are generalizations of well-known combinatorial optimization problems, but the generalization involves complicated side constraints that cannot be readily incorporated into optimization algorithms for the underlying standard problem. Thus, even if an optimization algorithm were available for the underlying problem, it might not be applicable to the problem at hand.

From an application point of view, approximation algorithms range from tailored algorithms, designed for a specific problem type, to general algorithms, which can be applied to a broad range of problem types. For the latter, it is often more appropriate to speak of algorithmic templates since in many cases the problem specific details still need to be filled in to obtain an operational algorithm. Local search is an example of such an algorithmic template for combinatorial optimization [Papadimitriou & Steiglitz, 1982; Yannakakis, 1991]. Traditionally, local search stands for iterative improvement with the notable exception of the variable-depth search algorithms proposed by Kernighan & Lin [Kernighan & Lin, 1971; Lin & Kernighan, 1973]. The introduction of simulated annealing, a randomized approach to local search [Kirkpatrick, Gelatt & Vecchi, 1983; Černy, 1985], and its successful application to problems in many different areas [Aarts & Korst, 1989; Van Laarhoven & Aarts, 1987], has broadened the scope of local search and has led to new algorithmic templates such as threshold accepting [Dueck & Scheuer, 1988], tabu search [Glover, 1989] and to variants of genetic algorithms [Goldberg, 1989]. Also certain classes of neural networks exhibit a strong relation with local search [Baum, 1986]. For a detailed overview the reader is referred to Aarts, Korst & Zwietering [1991].

For a number of these approaches theoretical results are available regarding the asymptotic convergence to global minima. However, only little is known about the theoretical finite-time performance of local search. Therefore, many authors have been carrying out computational studies in which the various approaches are applied to specific combinatorial optimization problems, trying to reveal their real strength, for instance in comparison with tailored optimization and approximation algorithms. For example, Johnson et al. [1989; 1991] compared simulated annealing with several other algorithms for the traveling salesman, graph coloring, graph partitioning and number partitioning problems. The general conclusion from these studies seems to be that the new approaches are eminently suitable to be applied to large instances of problems that are hard to model and for which no satisfactory tailored algorithms are available. Evidently, this area provides ample opportunity for a general approach whose implementation requires only a modicum of sophistication.

The present paper belongs to a series of papers in which we report on the computational performance of local search algorithms when applied to specific combinatorial optimization problems; in a previous paper [Ulder et al., 1990], we presented results for the traveling salesman problem [Lawler et al., 1985]. Here, we consider the job shop scheduling problem, one of the computationally more difficult combinatorial optimization problems [French, 1982; Lawler et al., 1992]. In addition to a mutual comparison, we also compared various local search approaches with a number of job shop scheduling heuristics known from the literature.

The organization of this paper is as follows. In Section 2 we define the job shop scheduling problem and describe the various neighborhoods used by the local search algorithms under investigation. Section 3 presents the algorithms compared in this paper and Section 4 contains the numerical results. The paper ends with some conclusions and final remarks.

2 Job shop scheduling

The job shop scheduling problem is defined as follows. Given are n jobs and m machines. Each job consists of a sequence of operations, which must be executed in a given order. Each operation has to be executed on a given machine for a given period of time. A machine can perform at most one operation at a time. The problem is to find a schedule, i.e., an assignment of the operations to time intervals, such that the makespan, given by the total length of the schedule, is minimal.

To apply local search, one has to define a set of feasible solutions, a cost function and a neighborhood structure. Here, we use the formulation of Roy & Sussmann [1964], who represent an instance of job shop scheduling by a disjunctive graph $G = (V, A, E)$, where the vertex set V corresponds to the set of operations, the arc set A consists of arcs connecting consecutive operations of the same

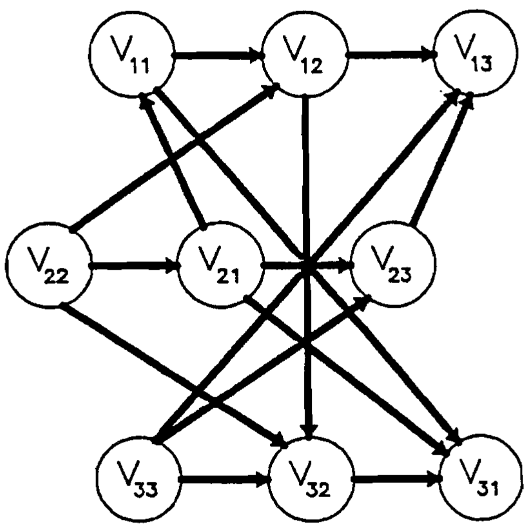

Figure 1. An example of a 3-job, 3-machine instance. Node v_{ij} corresponds to an operation of job i processed on machine j.

job, and the edge set E consists of edges connecting operations that must be executed on the same machine. Figure 1 gives an example for a 3-job 3-machine instance in which each job consists of three operations. The first job is to be carried out on machines 1, 2 and 3 (in that order); the second job on machines 2, 1 and 3, and the third one on machines 3, 2 and 1. Solid lines denote arcs and dotted lines denote edges. The weight of a vertex is given by the processing time of the corresponding operation. A feasible solution i can be defined as a set of orientations for the edges in E, such that the resulting digraph D_i is acyclic; the cost of a solution is then given by the length of the longest path in D_i. Figure 2 shows a solution for the instance of given in Figure 1. In this paper we consider the following two neighborhood structures.

(N_1) Given a solution i characterized by a digraph D_i, a neighboring solution is obtained by choosing two operations v and w that are immediate successors on some machine k and for which the arc (v, w) is on a longest path in D_i, and reversing (v, w) or, in other words, reversing the order in which v and w are processed on machine k. For this neighborhood the following results are of interest [Van Laarhoven, Aarts & Lenstra, 1992].

- (i) The reversal of (v, w) results in an acyclic digraph D_j, corresponding again to a feasible solution j.
- (ii) Reversals of arcs on the longest path are the only arc reversals that can (but need not) result in a digraph with a shorter longest path than the original digraph.
- (iii) For any digraph D_i, corresponding to an arbitrary solution i, it is possible to construct a sequence of arc reversals leading from D_i to a digraph corresponding to a globally minimal solution. This is a necessary and sufficient condition for asymptotic convergence of simulated annealing.

(N_2) Given a solution i and its digraph D_i, a neighboring solution is obtained by chosing again two vertices v and w satisfying the conditions mentioned under (1), but restricting the choice

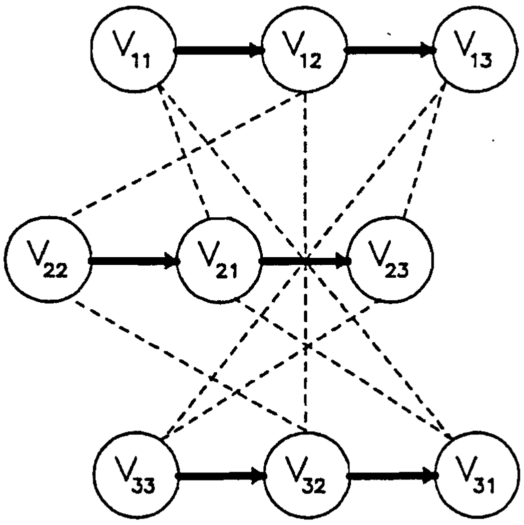

Figure 2. A solution to the instance of Figure 1.

to vertices for which at least one of the arcs $(p(v), v)$ and $(w, s(w))$ is not on a longest path, where $p(v)$ and $s(w)$ are the predecessor of v and the successor of w on machine k, respectively. This restriction is motivated by the fact that if both $(p(v), v)$ and $(w, s(w))$ are on a longest path, then the reversal of (v, w) cannot lead to a shorter longest path [Matsuo, Suh & Sullivan, 1988]. In addition to reversing (v, w), also $(p(u), u)$ and $(x, s(x))$ are reversed, where u is the immediate predecessor of w in the job sequence of w and x is the immediate successor of v in the job sequence of v. A detailed description of this neighborhood is given by Matsuo, Suh & Sullivan [1988]. Note that in this case the neighboring solutions of a solution i are obtained by reversing three arcs in D_i. Thus, this neighborhood is more intricate than the one given under (1). Finally we remark that (i) does not hold for this neighborhood since not all reversals pertain to arcs on a longest path. (ii) does not apply here and (iii) is open.

3 The algorithms

We now describe the algorithms used in the computational comparison. The multi-start iterative improvement and simulated annealing algorithms are only briefly mentioned; threshold accepting and genetic local search are presented in more detail.

3.1 Multi-start iterative improvement

Our multi-start iterative improvement algorithm consists of a number of cycles. Each cycle is started with a randomly generated solution and subsequently the neighborhood is searched for a better solution. If such a solution is found, it replaces the current one and the search continues in the neighborhood of the new solution. A cycle is terminated when a local minimum is reached, i.e. a solution whose cost is at least as good as that of all of its neighbors. After termination of a cycle a new one is started. This process continues until some specified limit on the running time (or a specified number of cycles) is reached. The best solution found in all cycles is returned as the final solution obtained by the algorithm.

3.2 Simulated annealing

Simulated annealing is a randomized version of local search. In addition to cost improving neighbors, which are always accepted, also cost deteriorating neighbors are accepted with a positive probability that gradually decreases in the course of the algorithm's execution. The lowering of the acceptance probability is controlled by a parameter, whose values are determined by a cooling schedule. Simulated annealing has been widely applied and in many cases it finds good quality solutions. For more details, the reader is referred to Aarts & Korst [1989] or Van Laarhoven & Aarts [1987]. A detailed study of simulated annealing for job shop scheduling is reported by Van Laarhoven, Aarts & Lenstra [1992].

Our simulated annealing algorithm uses the cooling schedule described by Van Laarhoven & Aarts [1987]. This is a three-parameter schedule, where the parameters χ_0 and ε_s determine the initial and final values of the control parameter, respectively, and the third parameter δ determines the decrement of the control parameter.

3.3 Threshold accepting

Threshold accepting, proposed by Dueck & Scheuer [1988], is a deterministic version of simulated annealing. A neighbor is accepted if the difference in cost between the neighbor and the current solution is smaller than a nonnegative threshold. The threshold values vary in the course of the algorithm's execution. Initially they are large and subsequently they are gradually decreased to become zero in the end. So far, no general rules are known that determine appropriate threshold values for an arbitrary optimization problem. This is in contrast to the many results on cooling schedules for simulated annealing. What is available though is the set of 30 threshold values that where successfully applied by Dueck & Scheuer to find solutions for a 442-city instance of the traveling salesman problem. Here we used the following empirical argument to find appropriate threshold values for job shop scheduling. Table 1 contains the average values of the makespans

μ	$\bar{f}$	μ	$\bar{f}$	μ	$\bar{f}$	μ	$\bar{f}$	μ	$\bar{f}$	μ	$\bar{f}$
1	1182.2	6	1002.6	11	1011.2	16	1005.8	21	1049.6	26	1034.8
2	1124.8	7	1007.8	12	987.4	17	1030.8	22	1032.0	27	1045.4
3	1047.0	8	1013.8	13	1023.6	18	1010.8	23	1022.4	28	1039.4
4	1124.8	9	1002.2	14	1012.0	19	1014.4	24	1048.0	29	1042.0
5	1022.2	10	990.6	15	1021.8	20	1044.8	25	1022.2	30	1040.6

Table 1. Average makespan $\bar{f}$ from 5 runs of threshold accepting for a 10-job, 10-machine job shop scheduling instance with 20 different sets of thresholds, each corresponding to a different value of the scaling parameter μ.

obtained from five runs of threshold accepting for a 10-job 10-machine instance obtained with 30 different sets of thresholds. At each threshold value, 2000 trials are allowed. The sets of threshold values are obtained by multiplying the values of Dueck & Scheuer by a factor $\mu \cdot \gamma$, where $\mu = 1, 2, \ldots, 30$ and γ is the average difference in cost of the cost increasing trials in a random walk of length 10,000 through the set of feasible solutions. Based on the results in Table 1 and similar results obtained for other instances, we decided to use $\mu = 12$ in our study. We do not rule out the possibility that fine tuning of the threshold values for job shop scheduling may yield better results.

3.4 Genetic algorithms

The effectiveness of multi-start iterative improvement may be improved by using the information available from the solutions obtained in the indivudual cycles. Several authors have proposed variants of local search algorithms, using ideas from population genetics; see for example, Mühlenbein, Gorges-Schleuter & Krämer [1987], Jog, Suh & Van Gucht [1989], and [Ulder et al., 1990]. We propose a genetic local search algorithm consisting of the following steps.

Step 1. Initialize: construct an initial population of P solutions.

Step 2. Improve: use local search to replace each solution by a locally optimal one; thus, a population of P local minima is obtained.

Step 3. Recombine: expand the current population by adding P offspring solutions obtained by recombining P pairs of solutions in the current population.

Step 4. Improve: use local search to replace each offspring solution by a locally optimal one; thus, a population of $2P$ local minima is obtained.

Step 5. Select: reduce the extended population to its original size by selecting the best P solutions.

Step 6. Iterate: repeat Steps 3 to 5 until a stopping criterion is satisfied.

We now discuss some of these steps in more detail.

Step 1. Initialization is done by randomly generating P solutions.

Steps 2&4. We use local search algorithms based on one of the neighborhood structures mentioned in Section 2.

Step 3. The candidate solutions for recombination are chosen randomly from the current population. The recombination of a pair of solutions i and j is based on the observation that the cost of a solution is given by the length of its longest path. Recombination is done by reversing some arcs on the longest path of j such that the resulting arcs are identical to those in i. More specifically, we choose, at random, an arc (v, w) in D_i. If the arc (w, v) occurs in D_j and belongs to a longest path in D_j, it is reversed. Next the longest path in the resulting digraph $D_{j'}$ is computed, and D_j is replaced by $D_{j'}$. Choosing an arc in D_i and reversing a corresponding arc in D_j is repeated k times. The net result is an offspring solution obtained from j by implanting a subset of arcs from D_i into D_j. In our implementation, k is set to $\lfloor n \cdot m/2 \rfloor$. Extensive experimentation reveals that the precise value of k is not very critical with respect to the effectiveness of the algorithms [Ulder, 1990].

Step 6. The algorithm terminates when either all solutions in the population have equal cost, or the best makespan in the population has not changed for K subsequent generations (i.e., repetitions of Steps 3 to 5). In our implementation we used $K = 10$.

4 Computational results

4.1 Comparison of local search algorithms

We compared the performance of the following seven algorithms: multi-start iterative improvement with neighborhoods N_1 and N_2 (MSII1 and MSII2, respectively), threshold accepting (TA), simulated annealing with N_1 and N_2 (SA1 and SA2, respectively), and genetic local search with N_1 and N_2 (GLS1 and GLS2, respectively). The comparison is carried out for a set of 43 instances. The first 40 instances are due to Lawrence [1984]; the remaining three to Fisher & Thompson [1963]. The last three include the notorious 10-job 10-machine instance that has defied solution to optimality for more than twenty years. For all instances, the number of operations of each job equals the number of machines and each job has precisely one operation on each machine.

The algorithms were programmed in PASCAL and care was taken to have identical data structures and subroutines wherever possible. For each instance, the algorithms were allowed about equal amounts of running time; differences of at most 2% occur, with the exception of the easy D, G and H instances. The reference point for each instance was given by the time taken by the SA-algorithm described in Section 3.2, with the N_1 neighborhood and with the parameters values $\chi_0 = 0.95$, $\delta = 1$, and $\varepsilon_s = 10^{-5}$, respectively (SA1). The multi-start algorithms were terminated when their running times exceeded those of SA1. For threshold accepting we tuned the number of trials per threshold and for genetic algorithms the population size to have about the same running times as SA1; the precise values of these parameters are given in Table 2. Finally, for SA2, we used $\delta = 1.15$ to equalize running times. The results of the comparison are displayed in Table 3. The

	A1-A5	B1-B5	C1-C5	D1-D5	F1-F5	G1-G5	H1-H5	I1-I5	FISH06	FISH10	FISH20
T	2100	3600	5000	6750	700	800	900	6250	500	2100	1600
P_1	55	45	33	33	25	30	25	60	45	35	20
P_2	40	35	25	25	20	25	20	45	35	25	13

Table 2. Number of trials per threshold T for threshold accepting, and population sizes P_1 and P_2 for genetic local search with N_1 and N_2 neighborhoods.

entries in the table are average makespans, computed from the solutions obtained by running each algorithm five times for each instance. From Table 3 we can make the following observations.

(*i*) Using a more intricate neighborhood structure pays off: MSII2 finds better solutions than MSII1 and GLS2 finds better ones than GLS1. For simulated annealing, however, the difference between SA1 and SA2 is not significant.

Instance	n	m	$\bar{f}$							$\bar{t}$
			MSII1	MSII2	TA	SA1	SA2	GLS1	GLS2	
A1	10	10	1029.6	1019.0	977.8	976.8	969.0	976.8	976.8	88.2
A2	10	10	858.4	829.2	829.8	787.2	785.6	791.2	791.6	91.4
A3	10	10	930.2	894.0	901.2	859.8	856.0	856.6	858.4	96.8
A4	10	10	922.6	899.6	925.8	855.0	854.6	863.6	859.0	93.8
A5	10	10	963.6	959.8	974.0	911.4	911.4	913.2	916.2	107.6
B1	15	10	1226.8	1181.2	1104.2	1083.6	1078.0	1084.4	1085.2	243.4
B2	15	10	1127.2	1099.2	979.6	950.2	950.2	954.0	944.0	254.2
B3	15	10	1164.8	1142.0	1033.7	1032.0	1032.0	1032.0	1032.0	242.2
B4	15	10	1111.8	1076.0	1014.6	962.0	960.4	970.0	981.2	234.8
B5	15	10	1167.8	1126.4	1075.4	1003.0	1019.6	1016.4	1010.0	254.8
C1	20	10	1473.2	1389.6	1227.0	1225.2	1221.2	1240.2	1236.2	469.6
C2	20	10	1502.4	1447.6	1289.8	1282.0	1275.6	1303.6	1300.8	492.0
C3	20	10	1486.4	1424.0	1286.4	1249.2	1242.8	1281.0	1264.8	455.6
C4	20	10	1485.2	1426.4	1262.4	1233.4	1225.8	1290.2	1260.0	471.0
C5	20	10	1619.2	1546.6	1366.6	1355.0	1355.0	1402.2	1386.0	441.4
D1	30	10	2013.0	1946.8	1784.0	1784.0	1784.0	1784.0	1784.0	879.8
D2	30	10	2119.4	2009.6	1850.0	1850.0	1850.0	1850.0	1850.0	909.0
D3	30	10	1958.4	1879.0	1719.0	1719.0	1719.0	1719.0	1719.0	1045.6
D4	30	10	2021.4	1937.8	1721.0	1721.0	1721.0	1737.0	1730.2	1055.4
D5	30	10	2158.4	2123.4	1888.0	1888.0	1888.0	1894.2	1890.2	1004.4
F1	10	5	670.4	666.0	682.4	666.0	666.0	666.0	666.0	17.2
F2	10	5	700.8	688.0	693.8	669.2	658.6	668.0	659.0	18.6
F3	10	5	645.0	633.2	628.2	617.6	616.6	613.2	609.2	21.0
F4	10	5	620.6	614.8	632.0	598.8	590.6	599.6	594.0	16.8
F5	10	5	593.0	593.0	593.0	593.0	593.0	593.0	593.0	13.6
G1	15	5	926.0	926.0	926.0	926.0	926.0	926.0	926.0	25.4
G2	15	5	923.0	915.0	894.0	890.0	919.8	890.0	890.0	42.2
G3	15	5	884.0	865.6	863.0	863.0	863.0	863.0	863.0	39.8
G4	15	5	952.6	951.0	951.0	951.0	951.0	951.0	951.0	35.6
G5	15	5	958.0	958.0	958.0	958.0	958.0	958.0	958.0	17.0
H1	20	5	1227.0	1222.4	1222.0	1222.0	1222.0	1222.0	1222.0	59.6
H2	20	5	1046.8	1039.0	1039.0	1039.0	1039.0	1039.0	1039.0	50.4
H3	20	5	1160.8	1151.6	1150.0	1150.0	1150.0	1150.0	1150.0	50.4
H4	20	5	1292.0	1292.0	1292.0	1292.0	1292.0	1292.0	1292.0	21.2
H5	20	5	1312.6	1284.4	1207.0	1207.0	1207.0	1207.0	1207.0	76.6
I1	15	15	1523.8	1456.6	1385.8	1307.8	1308.0	1324.8	1310.6	602.2
I2	15	15	1640.2	1598.8	1469.2	1440.8	1451.4	1449.4	1450.0	636.2
I3	15	15	1481.8	1421.8	1323.0	1235.4	1243.0	1285.2	1283.0	635.6
I4	15	15	1478.8	1445.4	1305.8	1258.2	1263.8	1279.0	1279.2	592.2
I5	15	15	1488.0	1439.0	1295.0	1256.4	1254.8	1273.2	1260.4	596.8
FISH06	6	6	56.2	55.0	60.4	55.0	55.0	55.0	55.0	9.4
FISH10	10	10	1055.8	1056.6	1003.8	969.2	977.8	978.2	982.2	99.4
FISH20	20	5	1356.0	1329.2	1228.6	1216.2	1245.0	1295.0	1294.2	88.4

Table 3. Average makespan $\bar{f}$ and running time $\bar{t}$ for multi-start iterative improvement (MSII), threshold accepting (TA), simulated annealing (SA), and genetic local search (GLS). The additions 1 and 2 correspond to the use of neighborhood N_1 or N_2, respectively; n and m denote the number of jobs and machines, respectively.

(*ii*) The multi-start algorithms are clearly inferior to the other algorithms. The difference becomes quite pronounced for the larger problem instances. For the D instances, for example, SA1 finds makespans that are on the average 12.8% shorter than those found by MSII1.

(*iii*) For those instances for which SA1 finds globally minimal solutions, i.e., the D, G, and H instances [Adams, Balas & Zawack, 1988], TA can compete with SA1. For the remaining, more interesting, instances, TA is clearly outperformed by SA1. The difference becomes again quite pronounced for the largest instances. For the I instances, SA1 finds makespans that are on the average 4.2% shorter than those found by TA.

(*iv*) The performance of SA1 and GLS2 is about equal, although for the largest instances SA1 consistently finds slightly better makespans. For the C instances, for example, SA1 is on the average 1.6% better and for the I instances 1.3%.

4.2 When time is of no concern

It is often argued that randomized variants of local search, such as simulated annealing and genetic local search, show their real strength when running times are of no concern. To verify this, we ran the algorithms SA1 and GLS2 for the ten tough instances identified by Applegate & Cook [1991]. Three of these instances are due to Adams, Balas & Zawack [1988]; the others are due to Lawrence [1984] and belong to the 43 instances considered in the previous section. For each instance, SA1 was run with parameter values $\chi_0 = 0.95$, $\delta = 10^{-4}$, and $\epsilon_s = 10^{-5}$; a single run took between two and 15 hours. GLS2 was given the same amount of time by tuning the population size. Table 4

Instance	n	m	$\overline{f}$				LB	status
			SA1	GLS2	A&C	B&C		
ABZ7	20	15	668*	680	668*	680	654	open
ABZ8	20	15	670*	698	687	701	635	open
ABZ9	20	15	691*	708	707	717	656	open
B1	15	10	1053*	1055	1053*	1071	1040	open
B4	15	10	935*	938	935*	953	935	solved
B5	15	10	983	985	977*	988	977	solved
C2	20	10	1249*	1265	1269	1256	1235	open
C4	20	10	1185*	1217	1195	1205	1120	open
I3	15	15	1208*	1248	1209	1226	1184	open
I5	15	15	1225	1233	1222*	1246	1222	solved

Table 4. Average makespan $\overline{f}$ when time is of no concern for simulated annealing (SA1), and genetic local search (GLS2) with the N_1 and N_2 neighborhoods, respectively, the shuffle algorithm of Applegate & Cook (A&C), the tabu search algorithm of Barnes & Chambers (B&C), the lower bounds from Applegate & Cook (LB), and the current status; a "" indicates the best known result.*

shows the results we obtained. For purposes of reference the table also includes the values found by Applegate & Cook's [1991] shuffle algorithm and by Barnes & Chambers' [1991] implementation of tabu search, as well as the lower bounds on the optimum found by Applegate & Cook's [1991] edge finder algorithm. Edge finder is a branch and bound algorithm, inspired by the work of Carlier & Pinson [1984], and shuffle is a heuristic variant of edge finder. For details we refer to the paper by Applegate & Cook.

It appears that, when time is of no concern, substantially improved results are obtained; c.f. Table 3. Furthermore, SA1 outperforms GLS2 even more clearly than before. SA1 also obtains generally better solutions than the approximation algorithms of Applegate & Cook and Barnes & Chambers, especially for the open problems. It should be taken into account, however, that their methods are much faster, but also much more tailored towards the specific structure of the job shop scheduling problem. In any case, SA1 has found better solutions than were previously known for five of the ten tough instances.

5 Discussion

We have compared a number of local search approaches by applying them to the job shop scheduling problem. From the computational results it can be concluded that the effectiveness of a standard iterative improvement algorithm can be improved either by relaxing the rigid concept of accepting cost improving solutions only, leading to threshold accepting and simulated annealing, or by using concepts from population genetics. This conclusion is in accordance with those drawn by Ulder et al. [1990] from a similar comparison for the traveling salesman problem, with the small but intriguing difference that, for job shop scheduling, threshold accepting is clearly outperformed by both simulated annealing and genetic local search, whereas for the traveling salesman problem threshold accepting and simulated annealing are about equally effective.

Finally, our experiments confirm that simulated annealing, when it is given enough time, can find better solutions than a number of faster and more tailored heuristics.

Bibliography

AARTS, E.H.L., J.H.M. KORST, P.J. ZWIETERING [1992], *Deterministic and Randomized Local Search*, Manuscript NR NL-M.S.17.235, Philips Research Laboratories, Eindhoven, The Netherlands.

AARTS, E.H.L., J.H.M. KORST [1989], *Simulated Annealing and Boltzmann Machines*, Wiley, Chichester.

ADAMS, J., E. BALAS, D. ZAWACK [1988], The shifting bottleneck procedure for job shop scheduling, *Management Science* **34**, 391-401.

APPLEGATE, D., W. COOK [1991], A computational study of the job-shop scheduling problem, *ORSA Journal on Computing* **3**, 149-156.

BARNES, J.W., J.B. CHAMBERS [1991], *Solving the Job Shop Scheduling Problem Using Tabu Search*, Technical Report ORP91-06, Graduate Program in Operations Research, University of Texas, Austin, USA.

BAUM, E.B. [1986], Towards practical "neural" computation for combinatorial optimization problems, in: J.S. Denker (ed.) *Neural Networks for Computing*, AIP Conf. Proc. **151**, Snowbird, UT, USA, 53-58.

CARLIER, J., E. PINSON [1989], An algorithm for solving the job shop problem, *Management Science* **35**, 164-176.

DUECK, G., [1990], private communication.

DUECK, G., T. SCHEUER [1990], Threshold accepting: a general purpose optimization algorithm, *Journal* *of Computational Physics* **90**, 161-175.

FISCHER, H., G.L. THOMPSON [1963], Probabilistic learning combinations of local job-shop scheduling rules, in: J.F. Muth, G.L. Thompson, eds., *Industrial Scheduling*, Prentice Hall, Englewood Cliffs, N.J., 225-251.

FRENCH, S. [1982], *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, Horwood, Chichester.

GAREY, M.R., D.S. JOHNSON [1979], *Computers and Intractability: a Guide to the Theory of NP-completeness*, Freeman and Company, San Francisco.

GLOVER, F. [1989], Tabu search - Part I, *ORSA Journal on Computing* **1**, 190-206.

GLOVER, F. [1989], Tabu search - Part II, *ORSA Journal on Computing* **2**, 4-32.

GOLDBERG, D.E. [1989], *Genetic Algorithms in Search, Optimization and Machines Learning*, Addison-Wesley, Reading (MA).

JOG, P., J.Y. SUH, D. VAN GUCHT [1989], The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem, *Proc. 3rd Int. Conference on Genetic Algorithms*, 110-115.

JOHNSON, D.S., C.R. ARAGON, L.A. MCGEOCH, C. SCHEVON [1989], Optimization by simulated annealing: an experimental evaluation; Part I, graph partitioning, *Operations Research* **37**, 865-892.

JOHNSON, D.S., C.R. ARAGON, L.A. MCGEOCH, C. SCHEVON [1991], Optimization by simulated annealing: an experimental evaluation; Part II, graph coloring and number partitioning, *Operations Research* **39**, 378-406. manuscript, Bell Laboratories, Murray Hill, USA.

KIRKPATRICK, S., C.D. GELATT JR., M.P. VECCHI [1983], Optimization by simulated annealing, *Science* **220**, 671-680.

LAARHOVEN, P.J.M. VAN, E.H.L. AARTS [1987], *Simulated Annealing: Theory and Applications*, Reidel, Dordrecht.

LAARHOVEN, P.J.M. VAN, E.H.L. AARTS, J.K. LENSTRA [1988], Job shop scheduling by simulated annealing, *Operations Research* **40**, 113-126.

LAWLER, E.L., J.K. LENSTRA, A.H.G. RINNOOY KAN, D.B. SHMOYS, EDS. [1985], *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, Chichester.

LAWLER, E.L., J.K. LENSTRA, A.H.G. RINNOOY KAN, D.B. SHMOYS [1989], *Sequencing and Scheduling: Algorithms and Complexity*, Designing Decision Support Systems Notes, NFI 11.89/63, Eindhoven University of Technology, Eindhoven, The Netherlands.

MATSUO, H., C.J. SUH, R.S. SULLIVAN [1988], *A controlled search simulated annealing method for the general job shop scheduling problem* , Working Paper 03-04-88, Graduate School of Business, The University of Texas at Austin, Austin, USA.

MÜHLENBEIN, H., M. GORGES-SCHLEUTER, O. KRÄMER [1988], Evolution algorithms in combinatorial optimization, *Parallel Computing* **7**, 65-85.

PAPADIMITRIOU, C.H., K. STEIGLITZ [1982], *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, N.J. USA.

ROY, B., B. SUSSMANN [1964], *Les problèmes d'ordonnancement avec constraints disjonctives*, Note DS No. 9 bis, SEMA, Paris, France.

ULDER, N.L.J. [1990], *Genetic Local Search: A Population-based Search Algorithm*, Technical Note NR 050/90, Philips Research Laboratories, Eindhoven, The Netherlands.

ULDER, N.L.J., E.H.L. AARTS, H-J. BANDELT, P.J.M. VAN LAARHOVEN, E. PESCH [1990], Genetic local search for the traveling salesman problem, *Lecture Notes in Computer Science* **496**, Springer, Berlin, 109-116.

YANNAKAKIS, M. [1990], The analysis of local search problems and their heuristics, *Lecture Notes in Computer Science* **415**, Springer, Berlin, 298-310.

List of COSOR-memoranda - 1992

Number	Month	Author	Title
92-01	January	F.W. Steutel	On the addition of log-convex functions and sequences
92-02	January	P. v.d. Laan	Selection constants for Uniform populations
92-03	February	E.E.M. v. Berkum H.N. Linssen D.A. Overdijk	Data reduction in statistical inference
92-04	February	H.J.C. Huijberts H. Nijmeijer	Strong dynamic input-output decoupling: from linearity to nonlinearity
92-05	March	S.J.L. v. Eijndhoven J.M. Soethoudt	Introduction to a behavioral approach of continuous-time systems
92-06	April	P.J. Zwietering E.H.L. Aarts J. Wessels	The minimal number of layers of a perceptron that sorts
92-07	April	F.P.A. Coolen	Maximum Imprecision Related to Intervals of Measures and Bayesian Inference with Conjugate Imprecise Prior Densities
92-08	May	I.J.B.F. Adan J. Wessels W.H.M. Zijm	A Note on "The effect of varying routing probability in two parallel queues with dynamic routing under a threshold-type scheduling"
92-09	May	I.J.B.F. Adan G.J.J.A.N. v. Houtum J. v.d. Wal	Upper and lower bounds for the waiting time in the symmetric shortest queue system
92-10	May	P. v.d. Laan	Subset Selection: Robustness and Imprecise Selection
92-11	May	R.J.M. Vaessens E.H.L. Aarts J.K. Lenstra	A Local Search Template (Extended Abstract)
92-12	May	F.P.A. Coolen	Elicitation of Expert Knowledge and Assessment of Imprecise Prior Densities for Lifetime Distributions
92-13	May	M.A. Peters A.A. Stoorvogel	Mixed H_2/H_∞ Control in a Stochastic Framework

Number	Month	Author	Title
92-14	June	P.J. Zwietering E.H.L. Aarts J. Wessels	The construction of minimal multi-layered perceptrons: a case study for sorting
92-15	June	P. van der Laan	Experiments: Design, Parametric and Nonparametric Analysis, and Selection
92-16	June	J.J.A.M. Brands F.W. Steutel R.J.G. Wilms	On the number of maxima in a discrete sample
92-17	June	S.J.L. v. Eijndhoven J.M. Soethoudt	Introduction to a behavioral approach of continuous-time systems part II
92-18	June	J.A. Hoogeveen H. Oosterhout S.L. van der Velde	New lower and upper bounds for scheduling around a small common due date
92-19	June	F.P.A. Coolen	On Bernoulli Experiments with Imprecise Prior Probabilities
92-20	June	J.A. Hoogeveen S.L. van de Velde	Minimizing Total Inventory Cost on a Single Machine in Just-in-Time Manufacturing
92-21	June	J.A. Hoogeveen S.L. van de Velde	Polynomial-time algorithms for single-machine bicriteria scheduling
92-22	June	P. van der Laan	The best variety or an almost best one? A comparison of subset selection procedures
92-23	June	T.J.A. Storcken P.H.M. Ruys	Extensions of choice behaviour
92-24	July	L.C.G.J.M. Habets	Characteristic Sets in Commutative Algebra: an overview
92-25	July	P.J. Zwietering E.H.L. Aarts J. Wessels	Exact Classification With Two-Layered Perceptrons
92-26	July	M.W.P. Savelsbergh	Preprocessing and Probing Techniques for Mixed Integer Programming Problems

Number	Month	Author	Title
92-27	July	I.J.B.F. Adan W.A. van de Waarsenburg J. Wessels	Analysing $E_k\|E_r\|c$ Queues
92-28	July	O.J. Boxma G.J. van Houtum	The compensation approach applied to a 2×2 switch
92-29	July	E.H.L. Aarts P.J.M. van Laarhoven J.K. Lenstra N.L.J. Ulder	Job Shop Scheduling by Local Search