A Comparison of Genetic Algorithms for the Static Job Shop Scheduling Problem

Manuel Vázquez and L. Darrel Whitley

Computer Science Department, Colorado State University, Fort Collins, CO 80523
vazquez,whitley@cs.colostate.edu

June 28, 2000

Abstract

A variety of Genetic Algorithms (GA's) for the static Job Shop Scheduling Problem have been developed using various methods: direct vs. indirect representations, pure vs. hybrid GA's and serial vs. parallel GA's. We implement a hybrid GA, called OBGT, for solving JSSP. A chromosome representation containing the schedule itself is used and order-based operators are combined with techniques that produce active and non-delay schedules. Additionally, Local Search is applied to improve each individual created. OBGT results are compared in terms of the quality of solutions against the state-of-the-art Nowicki and Smutnicki Tabu Search algorithm as well as other GAs, including THX, HGA and GA3. The test problems include different problem classes from the OR-library benchmark problems and more structured job-correlated and machine-correlated problems. We find that each technique, including OBGT, is well suited for particular classes of benchmark problems, but no algorithm is best across all problem classes.

1 Introduction

The goal in the static Job Shop Scheduling Problem (JSSP) is to find the sequence of j jobs to be completed on m machines such that the makespan (finish time of the last operation) is minimized. Davis [1985] was the first to use Genetic Algorithms to solve Job Shop Problems. Other heuristics such as Simulated Annealing (SA) and Tabu Search (TS), and exact techniques as branch and bound have also been used for solving JSSP. Jain and Meeran [1998, 1999] provide a good description of these techniques, including generic as well as problem specific strategies. The objective of this research is to evaluate the effectiveness of order-based operators combined with the Giffler and Thompson [1960] algorithm as a repair method. Results are compared against local search techniques

that incorporate domain-specific information about critical paths and critical blocks for solving JSSP.

The rest of this paper is organized as follows. In Section 2, a detailed description of the JSSP is given. Section 3 summarizes previous research related to our work. In Section 4, OBGT is sketched. Section 5 summarizes our results.

2 JSSP description

The JSSP is defined using the following notation :
$S = \{R, P, O, C\}$ is a 4-tuple representing the scheduling problem.
$R = \{R_1, R_2, ..., R_r\}$, representing the r resources or machines.
$P = \{P_1, P_2, ..., P_p\}$, representing the p products to be finished in the job shop. Each product $P_i = \{Op_{i1}, Op_{i2}, ..., Op_{iy}\}$ consists of a set of operations and an operational precedence relation (representing technological constraints).
$O = \{O_1, O_2, ..., O_o\}$ representing the orders flowing in the shop to finish the products. The orders in the job shop are represented for a tuple, $O_j = \{P_j, D_j\}$, where P_j is the product, and D_j is the due date of the order.
$C = \{C_1, C_2, ..., C_c\}$ representing the constraints of the problem.

The objective is to produce an ordered sequence of operations to process on each machine. The processing start times of each operation are assigned so as to optimize the objective function and to satisfy the problem constraints.

3 Previous research

3.1 Chromosome representations for JSSP

There are two ways of representing a schedule: indirect and direct. In *indirect* representations, the chromosome contains an encoded schedule. A schedule builder is used to transform the chromosome into a feasible schedule. Indirect representations range from traditional binary representations [Nakano, et al., 1991] to domain-specific knowledge representations [Bagchi, et al., 1991].

In *direct* representations, the chromosome directly represents the production schedule. Bruns [1993] and Yamada and Nakano [1992] describe many ways to deal with direct representations. Direct representation performs efficiently on production scheduling, incorporates domain-specific operations easily, but also requires domain-specific recombination operators.

3.2 Types of feasible schedules in JSSP

There are four classes of feasible schedules in JSSP: inadmissible, semi-active, active and non-delay. *Inadmissible* schedules contain excess idle time. It is possible to improve the scheduling quality by forward-shifting of operations until no excess idle time exists. *Semi-active* schedules contain no excess idle time, but they can be improved by shifting some operations to the front without delaying others. *Active* schedules contain no idle time and none of the operations can

be finished earlier without delaying other operations. The optimal schedule is guaranteed to be an active schedule. *Non-delay* schedules are active schedules, in which operations are placed into the schedule such that the machine idle time is minimized. No machine is kept idle if some operation can be processed.

Two methods are applied to generate feasible schedules. The Giffler and Thompson method (GT) which produces active schedules and the Non Delay algorithm(ND) which produces non-delay schedules. Both methods were originally described by Giffler et al. [1960]. In both methods, a heuristic is applied to select the next operation to be scheduled.

The Giffler and Thompson (GT) algorithm has been used in many JSSP implementations. Lin, Goodman and Punch [1997b] described the representations and the crossover operators used in previous GT-algorithm-based genetic algorithms for static JSSP. Usually, the GT algorithm is used to convert the offsprings into active schedules to guarantee feasibility.

Lin, Goodman and Punch [1997b] also created two operators inspired by the GT algorithm: THX crossover and THX mutation. They claim that these operators transmit temporal relationships present in the schedule. *THX crossover* selects a crossover point and uses it as a decision point in the GT algorithm to exchange information between two schedules. *THX mutation* builds the critical block for the scheduling and two operations in the block are randomly selected and reversed.

Hart and Ross [1998] used a representation called "Heuristic Combination Method" (HCM). HCM uses an implicit representation in which each gene in the chromosome contains a heuristic that performs the decision choice at each step during the schedule generation process. Their chromosome stores pairs *(method, heuristic)* where *method* is a choice between GT and ND and *heuristic* is the strategy followed to select the operation to be scheduled. This method, called the Heuristically guided GA (HGA), always produces active schedules. HGA was designed to deal with the dynamic JSSP, but is easily adapted to the static case. We previously replicated HGA results for dynamic JSSP benchmark problems [Vázquez, et al., 2000].

3.3 Hybrid methods and local search

JSSP is a hard problem that cannot be solved efficiently by applying any single technique and a great deal of research has focused on hybrid methods. Jain and Meeran [1998a] reviewed in detail Tabu Search, Genetic Algorithms and Simulated Annealing techniques and they proposed ways to produce hybrid solutions.

Jain, Rangaswamy and Meeran [1998b] describe and compare in detail JSSP neighborhood models and move evaluation strategies. The goal is to define a constrained neighborhood which can be evaluated quickly without loosing the ability to produce improved solutions.

Bagchi and coauthors [1991] discuss the use of a GA to rapidly reduce the search space to a size that can subsequently be handled by a deterministic search algorithm. Yamada and Nakano [1992], in their solution applicable to

large-scale Job Shop problems, proposed for future research that a combination of their technique with Local Search heuristics would produce improvements in their algorithm's performance.

3.4 Order-based operators for scheduling

We introduce a hybrid order-based GA, where order-based operators are combined with GT and ND methods. Our hypothesis is that the previous outstanding GAs for JSSP has been overly complicated. Order-based operators preserve the relative order of the operations to be scheduled, which is an important characteristic in scheduling problems. In the JSSP domain, order-based operators will not always produce feasible schedules. But the GT algorithm can be used to repair illegal offsprings.

Davis developed the first order-based operators that are well suited for many scheduling problems [Davis, 1985], [Davis, 1991].
a) Uniform Order-based Crossover : A number of elements are selected from one parent and copied to the offspring. The missing elements are taken in the order in which they appear in the other parent.
b) Order-based Mutation : A sub-list of elements is selected from the parent and this sub-list is randomly permuted.

Following Davis, Syswerda developed two sets of order-based operators that are well suited for many scheduling problems [Syswerda, 1991]. This includes a form of *order-based* crossover and mutation operators, as well as a *position-based* crossover and mutation operators. Syswerda's *position-based* crossover is the same as Davis' *Uniform Order-based Crossover.* Whitley and Yoo [1995] have shown that Syswerda's order-based and position-based operators are in fact identical in expected behavior. In the experimentation section, only the Davis' operator results are presented.

4 OBGT, An Algorithm for JSSP

We use permutations as a direct representation of the schedule. This allows for efficiently generating JSSP schedules and makes it easy to incorporate domain-specific operators. Figure 1 contains a description of our algorithm. Order-based operators are used to recombine the permutations and the Giffler and Thompson (GT) algorithm is used to repair illegal offsprings. This is similar to the strategy followed by Bierwirth, et al. [1999]. Fang [1994] indicates that optimal schedules are active schedules. As noted, the Giffler and Thompson (GT) algorithm generates active schedules and the Non Delay (ND) algorithm produces ND schedules. According to Fang, optimal schedules are not necessarily members of the subset of ND schedules for some JSSP instances. In our solution, the ND algorithm is used to generate half of the individuals of the initial population. The other half are GT active schedules.

We also incorporate domain-specific techniques (based on critical paths) into the GA to improve the performance of the solution. We implement the

Procedure *OBGT Algorithm*
S1 Generate Initial Population (50% ND, 50% GT)
S2 While not converged
S2.1 For all individuals from i = 1 to Popsize
S2.1.1 Cross string i with probability 'pc' (select 2nd parent randomly)
S2.1.2 Apply Mutation with probability 'pm' to both offsprings
S2.1.3 Apply GT algorithm to repair both offsprings
S2.1.4 Hillclimb both offsprings
S2.1.5 Replace worst individual with best offspring (if improvement exists)
S2.1.6 Sort population (bubble offspring into place)

Figure 1: OBGT Algorithm

Grabowski method described by Jain, et al. [1998b]. This neighborhood uses block structures, where a move is defined by repositioning an operation to either the front or the rear of a critical block.

Our Order-Based Giffler and Thompson (OBGT) for the static JSSP has the following characteristics:

- A direct chromosome representation, containing the schedule itself, is used. This includes a permutation of jobs for each machine. Each operation (i.e., job on a machine) contains its start time.
- Order-based operators were implemented.
- The GT and ND algorithms were utilized to generate active and non-delay schedules.
- Grabowsky critical path method was applied to each generated offspring
- A selection and replacement strategy similar to GENITOR [Whitley, et al., 1988] is used.

Notice from Figure 1 that our implementation is not exactly the GENITOR algorithm. Another description of our algorithm is given in [Vázquez, et al., 2000]. The population size, the probabilities of crossover and mutation were fixed to 100, .7 and .01 respectively. (Higher crossover rate seemed to work nearly as well, and it is unclear if using a pc < 1.0 is really necessary).

5 RESULTS

OBGT is compared against THX, a Tabu Search algorithm, Mattfeld's GA3 and HGA. We implemented the following algorithms : A serial version of Lin's

[1999b] THX, Nowicki and Smutnicki [1996] Tabu Search (NS), a serial version (using an integer representation instead of binary) of the Hart and Ross [1998] HGA. In the case of Mattfeld's GA3 [1996], we compared against his published results. NS Tabu Search and GA3 represent the respective state of the art for Tabu Search and Genetic Algorithms for solving JSSP.

In order to make fair comparisons all the algorithms are run 30 times with a population size of 100 and a maximum number of 100 generations. In the case of NS Tabu Search, the maximum number of iterations is set to 100,000.

5.1 Experiments

The classical Fisher and Thompson (FTO6 '6x6', FT10 '10x10', FT20 '20x5') Yamada and Nakano (YN1, YN2, YN3, YN4), Lawrence (LA21 '10x15', LA27 '20x10', LA29 '20x10', LA38 '15x15') and Taillard (TA051, TA052, TA053, TA054, TA055, TA056, TA057, TA058, TA059, TA060) benchmarks available in the OR-library were among the JSSP instances used. The solutions obtained by the different algorithms were compared against the best-known solutions [1].

All of the OR-library problems rely heavily on random generators. Assuming that 'real-world' JSSP instances contain certain amount of structure and additional constraints on the problem definition, a problem generator created by Watson and co-workers is used to generate eight machine-correlated problems (MC01a, MC02a, MC03a, MC04a, MC01b, MC02b, MC03b, MC04b) and eight job-correlated problems (JC01a, JC02a, JC03a, JC04a, JC01b, JC02b, JC03b, JC04b) [2]. In job-correlated problems, jobs have correlated processing times across machines. In machine correlated problems, job processing times are similar on a particular machine. A description of the problem generator can be found in [Watson, et al., 1999].

Prob.	BKS	NS			THX			HGA			GA3			OBGT		
		Best	Mean	sd	Best	Mean	sd	Best	Mean	sd	Best	Mean	sd	Best	Mean	sd
FT06	55	**55**	55.0	0.0	**55**	55.0	0.0	**55**	55.0	0.0	**55**	55.0	0.0	**55**	55.0	0.0
FT10	930	**930**	943.3	11.5	**930**	1003.6	22.0	935	945.4	7.7	**930**	943.7	6.6	**930**	965.4	10.3
FT20	1165	**1165**	1170.9	6.0	**1165**	1237.3	27.4	1179	1200.9	8.9	**1165**	1180.3	4.7	**1165**	1210.3	22.5

Table 1: Fisher and Thompson Benchmark Results

Tables 1-6 present the results for the benchmark problems. The Best-Known Solutions are denoted BKS. In the case of machine-correlated and job-correlated problems, the results are compared against the lower bounds (LB).

[1] The benchmark problems and best-known solutions can be found at http://mscmga.ms.ic.ac.uk/jeb/orlib/jobshopinfo.html

[2] This problem generator and the description of job-correlated and machine-correlated problems is available at http://www.cs.colostate.edu/sched/generator.html

Prob.	BKS	NS			THX			HGA			GA3			OBGT		
		Best	Mean	sd	Best	Mean	sd	Best	Mean	sd	Best	Mean	sd	Best	Mean	sd
YN01	888	897	916.4	10.6	930	977.1	15.2	939	948.1	5.2	904	911.9	4.5	**892**	952.9	13.4
YN02	909	918	934.3	9.8	958	987.8	23.1	952	963.2	7.6	928	940.5	5.6	**914**	947.5	21.7
YN03	893	905	924.7	10.2	953	980.3	23.7	931	942.1	6.1	907	918.8	7.3	**901**	951.7	19.4
YN04	968	982	1000.4	12.7	1012	1049.3	16.9	991	1002.3	6.2	992	1012.0	10.1	**972**	999.6	12.7

Table 2: Yamada and Nakano '20x20' Benchmark Results

Prob.	BKS	NS			THX			HGA			GA3			OBGT		
		Best	Mean	sd	Best	Mean	sd	Best	Mean	sd	Best	Mean	sd	Best	Mean	sd
LA21	1046	1056	1059.5	3.3	1075	1097.0	23.4	1099	1107.2	6.6	**1047**	1059.4	6.3	1062	1077.8	20.6
LA27	1235	1267	1282.5	8.5	1349	1370.8	28.0	1308	1318.5	6.2	**1236**	1261.6	5.1	1276	1291.6	18.3
LA29	1152	1191	1211.2	13.6	1333	1355.2	41.7	1296	1305.6	9.7	**1180**	1199.9	10.8	1205	1222.7	16.6
LA38	1196	**1196**	1211.3	15.7	1319	1337.3	29.9	1309	1318.4	7.6	1201	1222.5	12.2	1235	1251.3	13.4

Table 3: Lawrence Benchmark Results

5.2 Interpretation

In the Fisher and Thompson benchmark (Table 1), all the algorithms except HGA find the best-known solution. HGA is not able to find the best known solution for all problems, but in average is better than all of the algorithms except GA3. GA3 presents the best average solution and HGA behaves closely to GA3 in average.

In the Yamada and Nakano benchmark (Table 2), OBGT produces consistently the best solution, but in the average case GA3 is better. In this benchmark THX and HGA did not perform well and NS produces competitive solutions.

In the Lawrence series (Table 3), GA3 produced the best results in 3 out of 4 problems. NS is slightly superior to OBGT. HGA and THX did not produced good results and THX is not well suited for this benchmark.

In the Taillard benchmark (Table 4), NS outperforms the rest of the algorithms. HGA and GA3 produced relatively good results. OBGT and THX present a bad behavior.

For job-correlated problems (Table 5), the best algorithm is OBGT, but NS and HGA produced also good results. THX behaves well in $10x10$ instances and poorly in $20x20$ ones. NS was beaten in all instances except JC04a, JC03b and JC04b. JC04a is the easiest instance (since all the algorithms find the optimal solution).

For machine-correlated problems (Table 6), OBGT and NS performs the best (NS is slightly better). THX did not work well, although THX find the best solution for the MC03a case (but in the average case produces bad solutions) and HGA produced reasonable results.

Prob.	BKS	NS			THX			HGA			GA3			OBGT		
		Best	Mean	sd	Best	Mean	sd	Best	Mean	sd	Best	Mean	sd	Best	Mean	sd
TA51	2760	**2765**	2775.3	15.3	3004	3119.6	36.3	2858	2908.8	14.9	2870	2927.5	26.3	2947	3117.5	24.5
TA52	2756	**2756**	2767.3	22.2	3036	3126.9	34.9	2847	2889.0	16.3	2883	2932.9	26.4	2867	2992.5	30.3
TA53	2717	**2718**	2743.6	17.1	2860	2933.6	34.1	2770	2822.3	9.1	2742	2780.1	19.4	2836	2924.3	30.1
TA54	2839	2852	2862.1	16.8	2950	3054.2	41.2	2874	2909.4	15.7	**2839**	2852.3	14.2	2962	3065.5	26.5
TA55	2679	**2682**	2724.1	25.2	2929	3041.5	36.9	2798	2844.2	13.6	2798	2853.9	22.8	2940	3043.0	31.9
TA56	2781	**2849**	2860.2	23.1	3015	3111.9	31.1	2907	2937.9	12.3	2882	2920.8	23.4	3024	3103.1	26.5
TA57	2943	**2943**	2950.1	13.1	3116	3224.1	40.9	2971	3017.5	15.6	2989	3036.9	18.2	3129	3215.3	35.2
TA58	2885	**2896**	2918.2	18.1	3200	3252.0	33.6	2913	2994.1	12.9	2954	3001.3	21.1	3069	3149.8	29.7
TA59	2655	**2655**	2680.5	19.1	2922	3049.1	36.3	2745	2789.4	14.9	2742	2817.0	19.7	2968	3059.6	28.7
TA60	2723	**2723**	2742.1	17.1	2947	2995.2	34.1	2864	2886.9	8.5	2803	2826.8	11.3	2942	3019.8	26.7

Table 4: Taillard '30x20' Benchmark Results

Prob.	BKS	NS			THX			HGA			OBGT		
		Best	Mean	sd	Best	Mean	sd	Best	Mean	sd	Best	Mean	sd
JC01a	760	904	904	0	888	933.6	14.38	922	926.28	2.56	**885**	909.28	8.29
JC02a	1142	1292	1292.03	0.18	**1272**	1352.04	12.50	1303	1325.98	6.55	1304	1345.2	9.41
JC03a	1871	2142	2142	0	**2098**	2142.58	28.62	**2098**	2142.58	8.11	**2098**	2164.52	12.42
JC04a	2011	**2011**	2011	0	**2011**	2011.00	0	**2011**	2011.00	0	**2011**	2011.00	0
JC01b	1578	1801	1847.25	21.94	2057	2100.45	25.37	1887	1906.45	8.13	**1798**	1875.65	38.13
JC02b	2278	2613	2651.7	27.87	2870	2932.1	33.25	2689	2705.15	11.21	**2610**	2710.4	53.64
JC03b	2830	**3030**	3088.7	25.25	3426	3515.4	42.83	3214	3237.6	12.83	**3030**	3108.85	55.15
JC04b	4105	**4542**	4564.45	20.32	5034	5125.35	44.33	4648	4697.58	20.11	**4542**	4622.13	39.37

Table 5: Job Correlated Benchmark Results. JC01a, JC02a, JC03a and JC04a are '10x10' jobs. JC01b, JC02b, JC03b and JC04b are '20x20' jobs.

6 Conclusions

From the experimental section, it can be concluded that Order-Based operators combined with the Giffler and Thompson repair method (OBGT) is a viable alternative for solving JSSP. Results indicate that OBGT produced good results (except in the Taillard benchmark), but these results are disperse (high variance among makespan values). Future research needs to examine larger problems in order to determine the OBGT scalability.

Other important observation noticed from the experimentation section is that HGA is a really good near-optimal strategy and it does not reflect scalability problems. HGA consumes less computational resources than OBGT and the variance among HGA results is smaller than the variance obtained by the rest of the GAs (only GA3 produces less variance in some cases and HGA is competitive in terms of variance to NS). HGA is a good candidate to be applied in real-time scheduling and in dynamic scheduling applications.

During the evaluation for job-correlated and machine-correlated problems, OBGT produced the best results. Further research is necessary on problems that are more representative of real-world problems. NS Tabu Search did not produce the best results in job-correlated problems, but its performance was good on machine-correlated problems.

GA3 [Mattfeld, 1996] is a really competitive GA for solving JSSP, it only

Prob.	BKS	NS			THX			HGA			OBGT		
		Best	Mean	sd	Best	Mean	sd	Best	Mean	sd	Best	Mean	sd
MC01a	644	**736**	738.6	5.21	740	768.34	8.25	749	756.92	3.76	740	759.02	8.12
MC02a	785	**869**	869	0	894	917.94	13.38	882	907.20	5.04	**869**	895.02	9.02
MC03a	1234	1251	1252.27	2.07	**1247**	1317.94	36.50	1276	1309.46	7.66	1253	1290.52	9.79
MC04a	1494	**1599**	1601.8	1.86	1637	1676.58	27.29	1621	1640.48	6.21	1603	1629.58	12.38
MC01b	1505	1810	1820.6	12.69	2007	2036.35	15.32	1905	1910.7	2.65	**1803**	1850.3	20.12
MC02b	1600	1987	2015.7	15.21	2230	2275.55	23.62	2003	2008.3	2.39	**1985**	2030.3	18.67
MC03b	2007	**2233**	2241.3	22.2	2420	2492.45	36.13	2270	2299.5	9.13	2239	2343.5	39.22
MC04b	3097	**3201**	3240.5	15.28	3671	3752.7	39.15	3233	3253.2	9.37	3207	3288.8	41.62

Table 6: Machine Correlated Benchmark Results. MC01a, MC02a, MC03a and MC04a are '10x10' jobs. MC01b, MC02b, MC03b and MC04b are '20x20' jobs.

was beaten in the Yamada and Nakano benchmark. It will be interesting to test GA3 in job-correlated and machine-correlated problems.

According to Lin and colleagues [Lin, et al., 1997a], [Lin, et al., 1997b], THX produces good results in its parallel implementation (all of the results shown in this research considers a serial implementation). Hart and Ross HGA implementation uses a Parallel Island Model [Hart, et al., 1998]. GA3 is also a parallel GA and its design include several complicated issues (population flow in the diffusion model, inheritance of attributes, population entropy, etc.). Currently HGA is a simple sequential algorithm. We have also not been able to compare run times with algorithms such as GA3. Run times were kept minimal in the current experiments. A parallel island model of OBGT should reduce variance in the solution quality and perhaps allow it compete even better against algorithms such as GA3 on a broader range of problems.

7 Acknowledgments

Manuel Vázquez is a graduate student at Colorado State University supported by PDVSA under the program "Becas 1998-2000". Thanks to Jean-Paul Watson and Laura Barbulescu for the problem generator and code for Nowicki and Smutnicki's Tabu Search implementation. This effort was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-97-1-0271. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

Bagchi,S.; Uckum, S.; Miyabe, Y.; Kawamura, K. [1991]. "Exploring Problem-Specific Recombination Operators for Job Shop Scheduling". *International Conf. Genetic Algorithms (ICGA-91).*

C. Bierwirth and D. Mattfeld [1999]. "Production Scheduling and Rescheduling with Genetic Algorithms". Evolutionary Computation, Volume 7, Number 1: 1 -16, MIT

Press, 1999.
R. Bruns [1993]. "Direct Chromosome Representation and Advanced Genetic Operators for Production Scheduling". *International Conf. Genetic Algorithms (ICGA-93).*
L.Davis [1991]. "Order-Based Genetic Algorithm and the Graph Coloring Problem". In *Handbook of Genetic Algorithms*, Chapter 6.
L.Davis [1985]. "Job Shop scheduling with Genetic Algorithms". *International Conf. Genetic Algorithms (ICGA-85).*
H. Fang [1994]. Ph.D. Thesis. "Genetic Algorithms in Timetabling and Scheduling", Dept. Artificial Intelligence. University of Edinburgh, Scotland.
E. Hart and P. Ross [1998]. "A Heuristic Combination Method for Solving Job-Shop Scheduling Problems". *Parallel Problem Solving from Nature V.*
A. S. Jain, S. Meeran [1999]. "Deterministic Job-Shop Scheduling: Past, Present and Future". European Journal of Operations Research (to appear in Volume 118, Issue 2).
A.S.Jain, S. Meeran [1998a]. "A State-of-the-art review of Job-Shop Scheduling techniques". Submitted to Journal of Heuristics.
A.S.Jain, B. Rangaswamy, S. Meeran [1998b]. "Job Shop Neighborhoods and Move Evaluation Strategies". Journal of Scheduling. Submitted for Publication.
S. Lin, E.D. Goodman, W.F. Punch, III [1997a]. "A Genetic Algorithm Approach to Dynamic Job Shop Scheduling Problems". *International Conf. Genetic Algorithms (ICGA-97).*
S. Lin, E.D. Goodman, W.F. Punch, III [1997b]. "Investigating Parallel Algorithms on Job Shop Scheduling Problems". Evolutionary Programming Conference.
Nowicki, E. and Smutnicki, C. [1996]. "A Fast Taboo Search Algorithm for the Job-Shop Problem". Management Science, 42(6), 797-813.
D.C. Mattfeld [1996]. "Evolutionary Search and the Job Shop". Physica-Verlag, 1996.
Morton, T.E., and Pentico, D.W [1993]. "Heuristic Scheduling Systems". John Wiley and Sons, 1993.
R. Nakano, T. Yamada [1991]. "Conventional Genetic Algorithms for Job Shop Problems". *International Conf. Genetic Algorithms (ICGA-91).*
G. Syswerda [1991]. "Schedule Optimization Using Genetic Algorithms". *Handbook of Genetic Algorithms*, Chapter 21.
M. Vázquez and D. Whitley [2000]. "A comparison of Genetic Algorithms in solving the Dynamic Job Shop Scheduling Problem". Submitted to *Genetic and Evolutionary Computation Conference*, GECCO 2000.
J.P. Watson, L. Barbulescu, A.E. Howe and D. Whitley [1999]. "Algorithm Performance and Problem Structure in Flow-Shop Scheduling". 18^{th} National Conf. on Artificial Intelligence (AAAI-99).
D. Whitley and J. Kauth [1988]. "GENITOR: A different Genetic Algorithm". In Proc. Rocky Mountain Conf. on Artificial Intelligence.
D. Whitley and N. Yoo [1995]. Modeling Simple Genetic Algorithms for Permutation Problems. In *Foundations of Genetic algorithms-3.* Morgan Kaufmann.
T. Yamada and R. Nakano [1992]. "A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems". In Parallel Problem Solving from Nature 2 (PPSN 2).