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

Manuel Vázquez
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523 USA
vazquez@cs.colostate.edu

L. Darrell Whitley
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523 USA
whitley@cs.colostate.edu

Abstract

The majority of the research using evolutionary algorithms for the Job Shop Scheduling Problem (JSSP) has studied only the static JSSP. Few evolutionary algorithms have been applied to the Dynamic Job Shop Scheduling Problem (DJSSP) which is more similar to real-world applications. We implement a hybrid genetic algorithm for solving the dynamic job shop problem. A direct chromosome representation, containing the schedule itself, is used. Order-based operators are combined with techniques that produce active and non-delay schedules. We refer to our algorithm as the Order-Based Giffler and Thompson (OBGT) Genetic Algorithm. OBGT is compared in terms of the quality of solutions against published solutions for benchmark problems. OBGT consistently finds better solutions on larger problems compared to several other evolutionary algorithms, including Temporal Horizon GA (THX) and Heuristically guided GA (HGA).

1 Introduction

Automated scheduling technologies have a great impact in reducing operation costs and in increasing the operational autonomy of complex manufacturing systems. The goal of any scheduling system is to efficiently convert customer requirements into schedulable operations. Unfortunately, scheduling problems typically display multiple conflicting objectives. Jobs compete for resources that have to be allocated to satisfy the objectives in a highly constrained environment. In this work, we study Dynamic Job Shop Scheduling, where the goal is to find the sequence of j jobs to be completed on m machines such that a given objective function is minimized. There are a number of different objectives which could be minimized: 1) the total elapsed time to complete all jobs (makespan, used on static JSSP), 2) the weighted mean completion time, 3) the weighted mean lateness or 4) the weighted mean idle time.

The majority of the evolutionary computation research on the Job Shop Scheduling Problem (JSSP) has concentrated on static JSSP. In this version of the problem, all of the jobs are known in advance and no changes occur. In the dynamic job shop scheduling problem, new jobs can arrive after processing begins. Only a few evolutionary approaches ([Fang, 1994], [Lin, et al., 1997a], [Hart, et al., 1998] and [Bierwirth, et al., 1998]) have been applied to the dynamic case. The objective of this study is to evaluate the effectiveness of a genetic algorithm using order-based operators combined with the Giffler and Thompson algorithm as a repair method for solving dynamic job shop scheduling problems (DJSSP)

The rest of this paper is organized as follows. In Section 2, a detailed description of the Job Shop Scheduling Problem is given. Section 3 introduces the types of schedules existing in JSSP and presents two algorithms that produce two types of feasible schedules: active and non-delay. Section 4 summarizes previous genetic algorithm research aimed at solving the dynamic JSSP. Section 5 describes the application of order-based operators in scheduling problems. In section 6, the Order-Based Giffler and Thompson (OBGT) algorithm is introduced. Section 7 summarizes our results.

2 JSSP Description

A brief general description of the Job Shop Scheduling Problem is described below.

We define the following notation :
$S = \{M, J, C\}$ is a 3-tuple representing the problem.

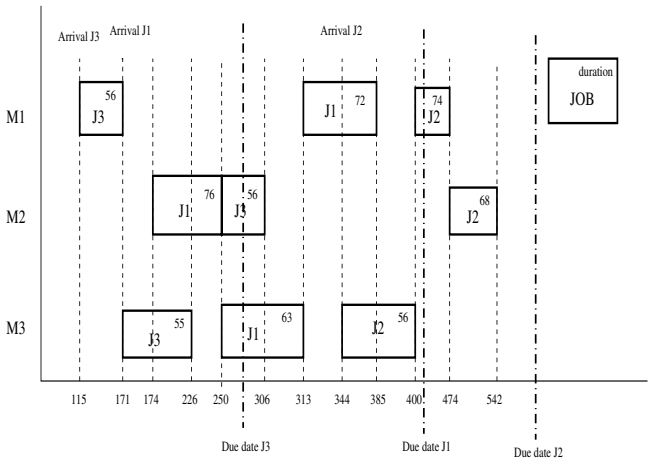

Figure 1: Schedule for the $3x3$ DJSSP described in Table 1

$M = \{M_1, M_2, ..., M_m\}$, representing the m machines.
$J = \{J_1, J_2, ..., J_j\}$, representing the j jobs.
Each job, $J_i = \{\{Op_{i1}, ..Op_{im}\}, at_i, dd_i, w_i\}$, consists of: a set of operations Op_{ij} specified by the processing time on each machine, the arrival time of the job at_i, the due date for the job dd_i, the job weight w_i and an operational precedence relation (preemption is not allowed). The arrival time is the time by which the job can start its processing. The due date is the time by which the job has to be completed. The job weight reflects the relative importance of the job in the scheduling process.
$C = \{C_1, C_2, ..., C_c\}$ representing the additional 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 desired objective function and to satisfy the problem constrains. Table 1 gives an example of a $3x3$ problem (3 jobs and 3 machines) and Figure 1 gives a schedule for this problem.

Table 1: Example of a $3x3$ DJSSP. For each job a sequence of operations is specified. (M) denotes machines, and [t] denotes processing times.

Job	Seq. of operations	Arr. time	Due date	Weight
J1	(2)[76] (3)[73] (1)[72]	174	430	1.80
J2	(3)[56] (1)[74] (2)[68]	344	568	3.93
J3	(1)[56] (3)[55] (2)[56]	115	296	2.24

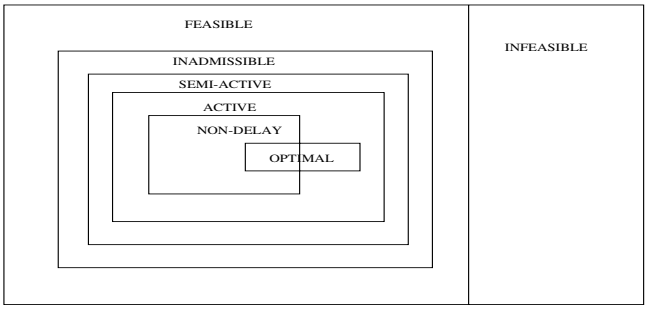

Figure 2: Relationship among the types of schedules in JSSP

2.1 JSSP Classification

In the *static JSSP* , all the jobs are ready to start at the beginning of the schedule and the main objective used is the makespan. In the *dynamic JSSP*, or *DJSSP*, jobs can arrive at some known *(deterministic)* or unknown *(stochastic)* future time. We present results for the deterministic DJSSP, because some of those we compare against require this information. However, our algorithm does not require or use information about future arrive times, and is a more general approach that also works for the stochastic DJSSP.

3 Feasible Schedules in JSSP

There are four classes of feasible schedules which apply to all JSSP variants: inadmissible, semi-active, active and non-delay. *Inadmissible* schedules contain excess idle time. It is possible to improve the schedule 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 some other operation. 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. Figure 2 shows the relationship among the types of schedules.

Hart and Ross [1998] applied two methods to generate feasible active schedules. The Giffler and Thompson method (GT), produces active schedules. The Non Delay algorithm (ND) produces non-delay schedules. Both algorithms, outlined in Figures 3 and 4, were originally described by Giffler et al., [1960].

```
Procedure Giffler and Thompson Algorithm
S1    build-Cset()
   /* Cset: set of all currently schedulable operations */
S2    tc = find-m*()
    /* m*, machine with minimum completion time tc */
S3    build-Gset()
         /* Set of conflicting jobs on m* */
         /* that can start before tc        */
S4    oper = select-job()     /* from Gset */
S5    delete-operation-from C()
S6    Repeat S1 to S5 until all operations are scheduled.
```

Figure 3: Giffler and Thompson Algorithm

```
Procedure Non Delay Algorithm
S1    build-Cset()
   /* Cset: set of all currently schedulable operations */
S2    build-Gset()
   /* Gset is the set of jobs that can start earliest */
S3    oper = select-job()     /* from Gset */
S4    delete-operation-from C()
S5    Repeat S1 to S4 until all operations are scheduled.
```

Figure 4: Non Delay Algorithm

In both methods, a heuristic is applied to select the next operation to be scheduled from the conflict set. Table 2 summarizes some of these heuristics. A good overview of heuristics for scheduling problems is given by [Morton, et al., 1993].

Table 2: Heuristics used in GT and ND, DJSSP case

Heuristic	Description
RND	Random
WSPT	Weighted Shortest Processing Time
WLWKR	Weighted Least Work Remaining
EOD	Earliest Operational Due Date
MST	Minimum Slack Time
FCFS	First Come First Served

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

4 Genetic Algorithms for DJSSP

The first publication using Genetic Algorithms (GAs) to solve the JSSP problem was by L. Davis [Davis, 1985]. Several researchers have published a variety of GAs for solving the static JSSP using various methods. These include GAs using direct representations vs. indirect representations, as well as hybrid Genetic Algorithms and parallel Genetic Algorithms [Bagchi, et al., 1991], [Nakano, et al., 1991], [Bruns, 1993] [Yamada, et al., 1992].

Few GA approaches have been developed to solve the DJSSP. Four GA approaches have been previously published: [Fang, 1994] [Lin, et al., 1997a], [Hart, et al., 1998] and [Biertwirth, et al., 1999]. All of the previous approaches, except Fang's algorithm, are based on the generation of active (Giffler and Thomson algorithm) and non-delay (ND) schedules.

Inspired by the GT algorithm Lin, Goodman and Punch [Lin, et al., 1997a], created two operators: THX crossover and THX mutation. They claim that these operators transmit temporal relationships present in the schedule. They looked at the time horizon defined for the whole schedule (Gantt Chart). In their representation, they encoded the operation starting times. The GT algorithm is used as an interpreter to transform any offspring into an active schedule. The *THX crossover* selects a crossover point (time in the Gantt Chart) and uses it as a decision point in the GT algorithm to exchange information between two schedules. The result is that the temporal relationships among operations in the parents are inherited by the offspring. *THX mutation* builds the critical block for the schedule and two operations in the block are randomly selected and reversed. The GT algorithm is applied to repair new offspring.

Hart and Ross [Hart, et al., 1998] used a representation called "Heuristic Combination Method" (HCM). HCM uses an implicit schedule 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. The chromosome length is the number of operations to be scheduled. Using the described representation, standard genetic operators are applied. This method, called the *Heuristically guided GA* (HGA), always produces active schedules.

Bierwirth and Mattfeld [Bierwirth, et al., 1999] created a GA strategy based on permutation operators and a permutation decoding method that produces semi-active, active and non-delay schedules. They used a total ordering permutation that determines the priority for each operation. During the selection of conflicting operations, the one with the minimum index

in the permutation list is chosen. They introduce a tuning parameter δ, which controls the period of time a machine is allowed to be idle. $\delta = 0$ produces non-delay schedules, while $\delta = 1$ produces active schedules. Intermediate δ values introduced a corresponding amount of delay.

Fang's algorithm [Fang, 1994] implicitly represents a schedule by encoding instructions for a schedule builder (heuristics are also incorporated in the chromosome). In Fang's approach, methods dealing with gene variance were applied to enhance the performance.

5 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 previous genetic algorithms for DJSSP have 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 DJSSP domain, order-based operators will not always produce feasible schedules. But the Giffler and Thompson algorithm can be used to repair illegal offsprings.

5.1 Sequencing Operators

Many researchers have described scheduling problems as a sequencing problem. Cleveland and Smith [1989] studied sequencing operators that do not preserve order information and which are also "blind" in the sense that they do not use domain-specific knowledge. As expected, these operators are not well suited for scheduling problems such as static and dynamic JSSP where preserving order information is critical. An empirical study conducted by Syswerda [1991] showed that the Edge recombination operator [Whitley et al., 1989], which produces excellent results for the Traveling Salesman Problem does not perform well in scheduling tasks. A study by Whitley et al, [1991] showed similar results. This motivated the use of order-based crossover operators.

5.2 Davis' Uniform Order-Based Operators

Davis developed the first order-based operators that are well suited for many scheduling problems [Davis, 1985], [Davis, 1991].
In this work, we used

a) Uniform Order-based Crossover : A number of elements are selected from one parent and copied to the offspring. The missing elements are taken from the other parent in order. This operator is shown in Figure 5.

```
    Parent 0:    8 6 4 2 1 5 9 3  7 10
    Parent 1:    2 3 4 6 7 1 5 9 10  8
    Template:    0 1 0 1 1 0 0 0  0  1

Offspring 1a:    _ 3 _ 6 7 _ _ _  _  8
     Missing:    4   2     1 5 9 10
 Offspring 1:    4 3 2 6 7 1 5 9 10  8

Offspring 0a:    8 _ 4 _ _ 5 9 3  7  _
     Missing:      2   6 1           10
 Offspring 0:    8 2 4 6 1 5 9 3  7 10
```

Figure 5: Davis' Uniform Order-Based Crossover. Assign "B" either 0 or 1. Let $\bar{B}$ denote the complement of B. Offspring Ba is constructed by selecting elements corresponding to B positions in the template from Parent B. The order of the missing elements for Offspring B are taken from Parent $\bar{B}$.

b) Order-based Scramble Mutation : A sub-list of elements is selected from the parent by selecting position x and y. The sublist of elements between positions x and y is randomly permuted.

Syswerda [1991] also developed two sets of order-based operators for scheduling problems. *a) Order-based Crossover* and *b) Position-based Crossover*. The position based crossover is the same as Davis' uniform order based crossover. Also, Whitley and Yoo [1995] proved that Syswerda's order-based crossover and position-based crossover are inverses of one another and that their behavior is identical in expectation.

6 Order-Based GT FOR DJSSP

We use a direct representation which contains the actual schedule. This allows for efficiently generating DJSSP schedules and makes it easy to incorporate domain-specific operators. Figure 6 contains a description of our algorithm. Order-based operators are used to recombine representations 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 sched-

Procedure *OBGT Algorithm*
S1 Generate Initial Population (50% ND, 50% GT)
S2 While not convergence
S2.1 For all individuals in the population $i = 1..n$
S2.1.1 Order-Based Crossover with 'P_c' (first parent is individual i, second parent random $U(1, n)$)
S2.1.2 Order-Based Mutation with 'P_m'
S2.1.3 Apply GT algorithm to repair both offsprings
S2.1.4 Replace the worst individual (individual n) with best offspring (if improve exists)
S2.1.5 Bubble the best offspring (if improve exists) to its position according to fitness

Figure 6: OBGT Algorithm

ules for some DJSSP 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. Our Order-Based Giffler and Thompson (OBGT) for the DJSSP 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., each job on a machine) contains its start time.
- Order-based crossover operators were implemented.
- The GT and ND algorithms were utilized to generate active and non-delay schedules along with the RND and WLWKR heuristics.
- A selection and replacement strategy similar to GENITOR [Whitley, et al., 1988] is used.

6.1 Algorithm Description

We apply $\mu + 1$ elitist selection. One individual is generated and replaces the worst member of the population, if an improvement occurs. There is no selection of parents. The members of the population get a chance to reproduce in turn (going from best to worst), with crossover probability 'P_c'. A mate is chosen randomly. As in the GENITOR algorithm (which is also a $\mu + 1$ algorithm), the offspring replaces the worst member of the population, and is bubbled into place to maintain a population sorted by fitness. Thus, population members may be "bumped" before reproducing. We also tested a GENITOR like strategy of using fitness biased selection and obtained similar results; only the results for the random strategy are reported.

Two different population sizes were studied, 100 and 300. Better results were obtained with a population size of 300 (improvements in the range from 1% to 5% were reached). The probabilities of crossover and mutation were fixed to 0.7 and 0.01 respectively and were not changed during the experimentation. (Higher crossover rate seemed to work nearly as well, and it is unclear if using a P_c less than 1.0 is really necessary.)

7 Results

OBGT is compared against Priority Dispatch Rules, Fang's algorithm, THX operators and HGA operators for solving benchmark dynamic scheduling problems. We were not able to compare with Biertwirth and Mattfeld algorithm because they did not publish results on the benchmark problems selected. Before presenting the results, the objective functions more commonly used in the DJSSP are described, along with our experimental setup.

7.1 Objective Functions

In static JSSP, the most common objective function used is makespan. Makespan is the finish time of the last operation. In dynamic JSSP, minimizing makespan is of less importance because the schedule horizon is open. For this reason, four different objective functions were tested. These functions are described by Lin, et al. [1997a] and also were used in HGA evaluation [Hart, et al., 1998].

Table 3 presents notation for various metrics used by the different objective functions. The objective functions are presented in Table 4. Notice that when the earliness (E_j) is incorporated in the ETWT objective function, a good schedule is obtained in such a way that the completion time of each job is equal to the due date (maybe not an active schedule).

Table 3: Notation used in the Objective Functions

w_j	Weight of job j
C_j	Completion time of job j
r_j	Release time of job j
d_j	Due date of job j
P_j	Processing time of job j
L_j	Lateness of job j=C_j-d_j
T_j	Tardiness of job $j = max(L_j, 0)$
E_j	Earliness of job $j = max(-L_j, 0)$

7.2 Experiments

Problem instances are of size 'JxM', where J is the number of jobs and M is the number of machines. Six algorithms are tested and compared on twelve bench-

Table 5: Weighted Tardiness Results TWT. Best known solutions are in bold fonts.

Problem	Size	PDR	Fang	THX-S	THX-P	HGA	OBGT
JB1	10x3	0.178	0.164	**0.162**	**0.162**	0.163	0.163
JB2	10x3	**0.086**	0.087	**0.086**	**0.086**	**0.086**	**0.086**
JB4	10x5	0.560	**0.556**	0.559	0.559	**0.556**	**0.556**
JB9	15x3	0.185	0.177	**0.169**	**0.169**	0.170	**0.169**
JB11	15x5	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
JB12	15x5	0.218	**0.139**	**0.139**	**0.139**	**0.139**	**0.139**
LJB1	30x3	0.276	0.215	0.224	**0.190**	0.192	0.201
LJB2	30x3	0.460	0.459	0.410	**0.395**	0.407	0.407
LJB7	50x5	0.109	0.110	0.090	0.060	**0.055**	**0.055**
LJB9	50x5	0.796	0.982	0.742	**0.615**	0.630	0.630
LJB10	50x8	0.479	0.455	0.478	0.438	0.418	**0.407**
LJB12	50x8	0.489	0.478	0.433	0.399	0.392	**0.342**

Table 6: Weighted Lateness Results LWT. Best known solutions are in bold fonts.

Problem	Size	PDR	Fang	THX-S	THX-P	HGA	OBGT
JB1	10x3	**-0.173**	-0.168	**-0.173**	**-0.173**	-0.167	**-0.173**
JB2	10x3	-0.812	-0.824	-0.816	**-0.839**	-0.819	-0.819
JB4	10x5	0.497	0.490	0.493	0.493	**0.489**	**0.489**
JB9	15x3	-0.047	-0.073	-0.078	-0.078	**-0.079**	**-0.079**
JB11	15x5	-0.607	-0.655	-0.730	**-0.751**	-0.702	-0.743
JB12	15x5	-0.082	-0.102	**-0.103**	**-0.103**	-0.102	-0.102
LJB1	30x3	-0.112	-0.195	-0.206	-0.214	**-0.224**	**-0.224**
LJB2	30x3	0.013	-0.043	-0.077	**-0.078**	**-0.078**	**-0.078**
LJB7	50x5	-0.411	-0.414	-0.453	-0.507	**-0.523**	**-0.523**
LJB9	50x5	0.622	0.702	0.498	**0.354**	0.388	0.388
LJB10	50x8	-0.038	-0.096	-0.082	-0.108	-0.139	**-0.143**
LJB12	50x8	0.256	0.221	0.192	0.113	0.121	**0.086**

Table 4: Weighted Objective Functions. Flow Time (FWT), Tardiness (TWT), Lateness (LWT), Earliness + Tardiness (ETWT)

O.F.	Definition(Normalized)
FWT	$(\sum_j w_j(C_j - r_j))/(\sum_j w_j P_j)$
TWT	$(\sum_j w_j T_j)/(\sum_j w_j P_j)$
LWT	$(\sum_j w_j L_j)/(\sum_j w_j P_j)$
ETWT	$(\sum_j w_j(E_j + T_j))/(\sum_j w_j P_j)$

mark problems ranging from '$10x3$' to '$50x8$' taken from Morton et al. [1993]. In the next paragraph, the algorithms are summarized.

Priority Dispatch Rules (PDR) algorithms apply a simple deterministic rule during the operation selection phase of the Giffler and Thompson algorithm. THX operators created by Lin et al. [1997] transmit the temporal relationships present in parents to the offsprings. Two versions of the THX algorithm (S, serial and P, parallel) are compared. Fang's [1994] algorithm works with an indirect representation and applies a schedule builder. The HGA method proposed by Hart et al. [1998] searches for the optimal schedule in the heuristics space. Our Order-Based GT (OBGT) algorithm combines order-based operators with the GT algorithm to repair the schedules.

The results for THX-S, THX-P and PDR were taken from Lin et al. [1997a]. (We implemented THX-S, but were unable to reproduce similar results.) The results for HGA and Fang were taken from Hart et al. [1998]. In tables 5-8, the results of the algorithms using the four different objective functions tested are shown. The results reflects the best over ten different runs. To be consistent with Lin et al. [1997a] and Hart et al. [1998], we ran our algorithm for 1,000 generations. OBGT always converged in less than 1,000 generations. Hart and Ross [1998] present four different versions of HGA; we report the best result obtained by any of the HGA version on each problem.

The best results during experimentation were obtained by THX-P, HGA and OBGT algorithms. However, pairwise t-test failed to show any difference between any of these algorithms. The t-test results are shown in Table 9. This is not surprising. Best known solutions are relatively common, and there is a ceiling effect because the results are near optimal. Thus, the data distribution is not normal, but rather log-normal. Another way to evaluate these algorithms is to compare the best solution found by each algorithm using a Chi-square test (Tables 10 and 11). The rows of the Chi-square table ($2x2$) represent the algorithm compared and the columns contain the number of times that the best-known solution is found or not for each problem.

The Chi-square test results in table 10 and 11 indicates

Table 7: Weighted Flow Time FWT. Best known solutions are in bold fonts.

Problem	Size	PDR	Fang	THX-S	THX-P	HGA	OBGT
JB1	10x3	**1.231**	1.237	**1.231**	**1.231**	1.236	**1.231**
JB2	10x3	1.772	1.778	1.768	1.768	**1.766**	**1.766**
JB4	10x5	1.111	1.109	1.108	1.108	**1.107**	**1.107**
JB9	15x3	1.947	1.768	1.754	1.754	1.754	**1.753**
JB11	15x5	1.795	1.794	1.723	**1.706**	**1.706**	**1.706**
JB12	15x5	1.257	1.259	**1.256**	**1.256**	**1.256**	**1.256**
LJB1	30x3	1.494	1.431	1.424	1.391	**1.389**	**1.389**
LJB2	30x3	1.924	1.826	1.783	**1.777**	**1.777**	**1.777**
LJB7	50x5	1.692	1.669	1.623	1.557	**1.542**	**1.542**
LJB9	50x5	2.490	2.659	2.475	**2.324**	2.337	2.341
LJB10	50x8	1.776	1.728	1.731	1.697	1.673	**1.663**
LJB12	50x8	2.207	2.138	2.115	2.080	2.058	**2.007**

Table 8: Weighted Earliness + Tardiness ETWT. Best known solutions are in bold fonts.

Problem	Size	PDR	Fang	THX-S	THX-P	HGA	OBGT
JB1	10x3	0.529	0.475	**0.474**	**0.474**	**0.474**	**0.474**
JB2	10x3	0.901	0.758	0.690	**0.499**	0.753	0.753
JB4	10x5	0.622	0.620	0.621	0.621	**0.619**	**0.619**
JB9	15x3	0.395	0.384	**0.369**	**0.369**	0.370	**0.369**
JB11	15x5	0.415	0.263	**0.262**	**0.262**	0.271	**0.262**
JB12	15x5	0.494	0.247	**0.246**	**0.246**	0.247	0.247
LJB1	30x3	0.654	0.322	0.322	**0.279**	0.280	**0.279**
LJB2	30x3	0.868	0.632	0.627	0.601	**0.598**	**0.598**
LJB7	50x5	0.515	0.374	0.345	0.254	0.269	**0.246**
LJB9	50x5	0.935	1.178	0.833	**0.739**	0.797	0.819
LJB10	50x8	0.882	0.621	0.688	0.598	0.548	**0.512**
LJB12	50x8	0.667	0.607	0.584	0.461	0.466	**0.399**

Table 9: Two sample T-Test. 95% Confidence Interval.

Algorithm	N	Mean	St.Dev.
THX-P	48	0.550	0.732
HGA	48	0.556	0.729
OBGT	48	0.551	0.730

ALGORITHMS	T	P
THX-P vs. HGA	0.04	0.97
THX-P vs. OBGT	-0.0	1.0
HGA vs. OBGT	0.03	0.97

Table 10: Chi-squared test, pair based, all problems (* indicates that the distribution is significant)

Comparison	X^2	$p \leq$
THX-P vs. OBGT	4.55	0.05(*)
HGA vs. OBGT	10.97	0.001(*)
HGA vs. THX-P	1.50	1

that the number of times that the OBGT finds the best-known solution is significant better than the rest of the algorithms. The results does not show difference between HGA and THX-P. Additionally, we note the following interesting points:

- In the two largest problems (LJB10 and LJB12), for all the objective functions, OBGT found a *new* best-known solution.
- In problem LJB9, THX-P always found the best solution.
- The best performance of OBGT is obtained for the FWT (Flow Time) objective function. In this function, OBGT is only outperformed by THX-P in problem LJB9.

Table 11: Chi-squared test, pair based, larger problems (* indicates that the distribution is significant)

Comparison	X^2	$p \leq$
THX-P vs. OBGT	5.48	0.025(*)
HGA vs. OBGT	8.39	0.01(*)
HGA vs. THX-P	0.35	1

8 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 DJSSP. The experimental section also suggests that OBGT behaves well in larger problems. This indicates that the current benchmark problem are perhaps too small and easy to solve ($50x8$ is considered a small problem in the JSSP domain). It is necessary to further investigate OBGT performance on larger problem instances and on more realistic test problems. Use of a parallel genetic algorithm greatly enhanced the THX results; but THX-P also used approximately 10 times more evaluations than OBGT. It

would also be very easy to implement a parallel version of OBGT using an Island Model. We expect this would further improve on the current results.

Acknowledgments

Manuel Vázquez is a graduate student at Colorado State University supported by PDVSA under the Program "Becas 1998-2000". This work was also supported by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-97-1-0271. The U.S. Government is authorized to reproduce and distribute reprints notwithstanding any copy notation thereon. We greatly appreciate the collaboration of Dr. Adele Howe and the rest of the members of the Scheduling Group at CSU.

References

[Bagchi, et al., 1991]: Bagchi,S.; Uckum, S.; Miyabe, Y.; Kawamura, K.: "Exploring Problem-Specific Recombination Operators for Job Shop Scheduling". In International Conference of Genetic Algorithms (ICGA) 91.

[Bierwirth, et al., 1999]: C. Bierwirth and D.C. Mattfeld. "Production Scheduling and Rescheduling with Genetic Algorithms". Evolutionary Computation, Volume 7, Number 1 : 1 -17, MIT Press, 1999.

[Bruns, 1993]: R. Bruns: "Direct Chromosome Representation and Advanced Genetic Operators for Production Scheduling". In International Conference of Genetic Algorithms (ICGA) 93.

[Cleveland, et al., 1989]: G.A. Cleveland, S.F. Smith: "Using Genetic Algorithms to Schedule Flow Shop Releases". In International Conference of Genetic Algorithms (ICGA) 89.

[Davis, 1991]: L.Davis. "Order-Based Genetic Algorithm and the Graph Coloring Problem". In Handbook of Genetic Algorithms, Chapter 6.

[Davis, 1985]: L.Davis "Job Shop scheduling with Genetic Algorithms". In International Conference of Genetic Algorithms (ICGA) 85.

[Fang, 1994]: H. Fang. Ph.D. Thesis. "Genetic Algorithms in Timetabling and Scheduling", Department of Artificial Intelligence. University of Edinburgh, Scotland.

[Giffler, et al., 1960]: Giffler, B and Thompson, G. "Algorithms for solving production scheduling problems". Operations Research, 8:487-503.

[Hart, et al., 1998]: E.Hart, P.Ross. "A Heuristic Combination Method for Solving Job-Shop Scheduling Problems". In Parallel Problem Solving from Nature V (PPSN V).

[Lin, et al., 1997a]: S.Lin, E.D.Goodman, W.F.Punch, III: "A Genetic Algorithm Approach to Dynamic Job Shop Scheduling Problems". In International Conference of Genetic Algorithms (ICGA) 97.

[Lin, et al., 1997b]: S.Lin, E.D.Goodman, W.F.Punch, III: "Investigating Parallel Algorithms on Job Shop Scheduling Problems". Sixth Annual Conference on Evolutionary Programming.

[Morton, et al., 1993]: Morton, T.E., and Pentico, D.W. "Heuristic Scheduling Systems", John Wiley and Sons, 1993.

[Nakano, et al., 1991]: R.Nakano, T.Yamada: "Conventional Genetic Algorithms for Job Shop Problems". In International Conference of Genetic Algorithms (ICGA) 91.

[Syswerda, 1991]: G.Syswerda: "Schedule Optimization Using Genetic Algorithms". In Handbook of Genetic Algorithms, Chapter 21.

[Whitley, et al., 1995]: D. Whitley and N. Yoo: "Modeling Permutation Encodings in Simple Genetic Algorithm". In Foundations of Genetic Algorithms (FOGA) 3.

[Whitley, et al., 1991]: T. Starkweather, D. Whitley, K.E. Mathias and S. McDaniels: "Sequence Scheduling with Genetic Algorithms". In New Directions for Operations Research in Manufacturing. Springer-Verlag

[Whitley, et al., 1989]: D.Whitley, T.Starkweather, D.Fuquay: "Scheduling Problems and Traveling Salesmen: The Genetic Edge recombination operator". In International Conference of Genetic Algorithms (ICGA) 89.

[Whitley, et al., 1988]: D. Whitley and J. Kauth: "GENITOR: A different Genetic Algorithm". In Proc. Rocky Mountain Conf. on Artificial Intelligence.

[Yamada, et al., 1992]: T.Yamada, R.Nakano: "A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems". In Parallel Problem Solving from Nature 2 (PPSN 2).