Directing the Search of Evolutionary and Neighbourhood-Search Optimisers for the Flowshop Sequencing Problem with an Idle-Time Heuristic

Peter Ross and Andrew Tuson

Department of Artificial Intelligence, University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, U.K.
{peter,andrewt}@dai.ed.ac.uk

Abstract. This paper presents a heuristic for directing the neighbourhood (mutation operator) of stochastic optimisers, such as evolutionary algorithms, so to improve performance for the flowshop sequencing problem. Based on idle time, the heuristic works on the assumption that jobs that have to wait a relatively long time between machines are in an unsuitable position in the schedule and should be moved. The results presented here show that the heuristic improves performance, especially for problems with a large number of jobs. In addition the effectiveness of the heuristic and search in general was found to depend upon the neighbourhood structure in a consistent fashion across optimisers.

1 Introduction

One problem of interest in the artificial intelligence and operations research scheduling communities is the flowshop sequencing problem. The aim of this problem is, quite simply, to find a sequence of n jobs that minimises the *makespan* — the time for all of the jobs to be processed by m machines. This paper will examine two ways how domain-knowledge can be included into evolutionary and stochastic neighbourhood search techniques: in the neighbourhood structure, and in biasing the order in which different possible moves (mutations) are tried to examine whether such knowledge is transferable between techniques; with an emphasis upon the second of these aspects.

The rationale for the second aspect is that, for each solution in the search space, there are number of 'moves' to other solutions available, most of which are non-improving. Conventional implementations examine these moves in a random order; however; if a method existed that could cheaply determine which moves were improving, it would then result in high-quality solutions being found more quickly (assuming that the search would not be led into a local optimum).

In practice, such a method would be heuristic in nature. Fortunately, it is often possible to examine a solution and to form beliefs about which moves would be most likely to improve the solution. These beliefs can then be captured in a heuristic, which makes it more likely that such moves are attempted first. The approach used here was based on the 'directed mutation' operator [15] that has been successfully used in timetabling, where exams with a high number of constraint violations were more likely to be moved; it is also related to the work by [18,19] on iterative repair scheduling, and by [9] on the 'min-conflicts' heuristic.

For this problem, such a belief can be formulated by using the idle times, the time a job spends waiting between machines, associated with each job in the sequence to decide what the most profitable moves are likely to be. Therefore, the effectiveness (or otherwise) of using the idle times as a heuristic to direct the search of some stochastic optimisation techniques will be examined. It should be noted that the idea of using idle times to improve the performance of heuristic methods for the flowshop sequencing problem is not new: [5] proposed the use of a similar heuristic in an approach that is essentially a deterministic hillclimber, and [13] used idle times in a constructive algorithm. Also, the concept of a 'candidate list', which orders the sequence in which the moves are considered has been used by [11], and in the evolutionary computing literature by [17].

However, the authors are unaware of an idle-time heuristic being used for evolutionary and stochastic neighbourhood search optimisation approaches, in this way, for this problem and whether, or to what degree, experience of representing domain knowledge gained with one optimisation technique can be transferred to another. Thus these will be main topics under investigation in this paper. The problem and the optimisers that are the subject of this investigation will be outlined first. The heuristic under investigation and how it is used to direct the search will then be described, and then the results obtained evaluated across a range of optimisers, and finally conclusions drawn.

2 The Flowshop Sequencing Problem

The flowshop sequencing, or $n/m/P/C_{max}$, problem involves finding a sequence of jobs for the flowshop (a straight line of machines to process, so as to minimise the *makespan* — the time taken for the last of the jobs to be completed. The makespan is dependent upon the order in which the products are produced because poor sequences can lead to temporary blockages in the flowline because completed stages cannot proceed, as a stage ahead in the flowline is still being processed (Figure 1).

This task is known to be NP-hard [7] (the number of possible sequences is $n!$) and can be formalised as follows: n jobs have to be processed (in the same order) on m machines; the aim is to find a job permutation $\{J_1, J_2, ..., J_n\}$ so as to minimise C_{max}. This is defined as follows: given processing times $p(i, j)$ for job i on machine j and the job permutation above, we can find the completion times by the following equations:

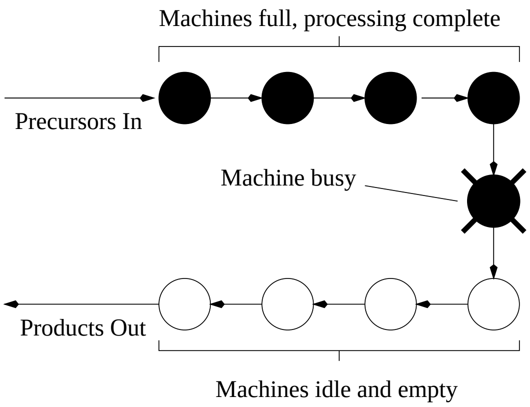

Fig. 1. Blocking of a Flowshop at a Single Machine

$$
\begin{aligned}
&C(J_1,1) = p(J_1,1)\\
&C(J_i,1) = C(J_{i-1},1) + p(J_i,1) \;\; for\ i = 2,...,n\\
&C(J_1,j) = C(J_1,j-1) + p(J_1,j) \;\; for\ j = 2,...,m\\
&C(J_i,j) = max\{C(J_{i-1},j), C(J_i,j-1)\} + p(J_i,j) \;\; for\ i = 2,...,n;\ j = 2,...,m\\
&C_{max} = C(J_n,m)
\end{aligned}
$$

Standard benchmarks exist for this problem, and were used in this study. Taillard [16] produced a set of test problems which, using a very lengthy Tabu Search procedure, were still inferior to their lower bounds, with problem sizes ranging from 20 jobs and 5 machines, to 500 jobs and 20 machines. There are 10 instances of each problem — all processing times were generated randomly from a $U(1,100)$ distribution.

3 The Optimisation Techniques Considered

A variety of stochastic optimisation methods (an overview is given in [4]) were examined: Stochastic Hillclimbing (SHC), Simulated Annealing (SA) [8], Threshold Accepting (TA) [3], Record-to-Record Travel (RTRT) [2], and an Evolutionary Algorithm (EA) [6]. Basic implementations were used in this study as the aim was to investigate whether the neighbourhood could be usefully directed, and not to find the most effective optimiser.

The encoding used for this problem was a permutation of the jobs to be placed into the flowshop, numbered 1 to n. Two *unary* neighbourhood (ie. mutation) operators were used in this study: permutation-shift, and permutation-swap (Figures 2 and 3), where the jobs that are bounded by a rectangle are the two jobs selected to be moved by the operator.

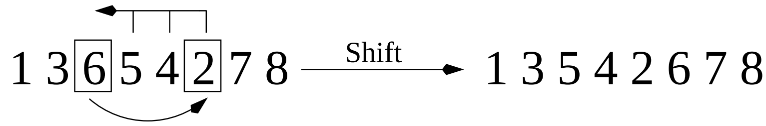

Fig. 2. The Permutation-Shift Operator

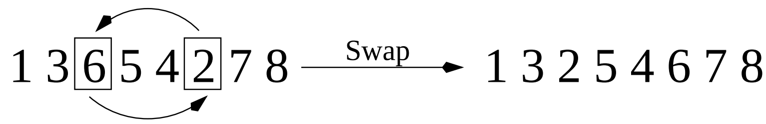

Fig. 3. The Permutation-Swap Operator

It should be noted that although the operation $swap(i, j)$ is equivalent to $swap(j, i)$, the same is not the case for $shift(i, j)$. In the case of the $shift(i, j)$ operator, position i (which is job number 6 at position 3 in the example above) corresponds to the job that is displaced to make room for the block of jobs that is to be shifted, the extent of which is given by position j. Exchanging the values of i and j in a shift operation therefore leads to a reversal of the roles of the two positions, and of the direction that the block is shifted.

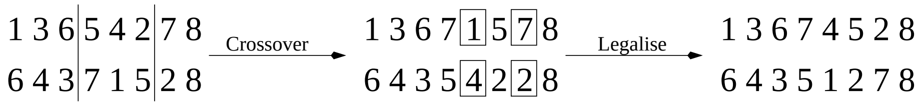

Fig. 4. The Modified-PMX Crossover Operator

The crossover operator used for the EA was 'Modified PMX' [10], found to give good results on this problem. This operator performs a two-point crossover upon the two strings selected. The repair procedure then analyses one string for duplicates: when one is found it is replaced by the first duplicate in the second string. This is repeated until both strings are legal (Figure 4).

4 The Idle Time Heuristic

The heuristic used here is based upon idle times. Idle time is usually defined for a particular job and machine, and is the amount of time that a job is waiting to be processed at a machine after it has been processed by the previous machine. This is illustrated by the Gantt chart (Figure 5) below — the gaps between blocks are the periods which a job is lying idle. However, as we are making a decision about which job to move, it would be sensible to add up the idle time for each job, over all machines, as given in the equations below:

$$I(J_1) = \sum_{j=2}^{m} p(J_1, j-1) \ (for\ the\ purposes\ of\ this\ study\ \ -\ \ see\ below)$$

$$I(J_i) = \sum_{j=2}^{m} max\{C(J_{i-1}, j) - C(J_i, j-1), 0\} \ for\ j = 2, ..., m$$

with processing times $p(i, j)$ for job i on machine j, the job permutation $\{J_1, J_2, ..., J_n\}$, completion time $C(J_i, j)$ and idle time $I(J_1)$.

Note that the first job in the sequence is given an idle time equal to the sum of the processing times on the machines $2, ..., n$ for that job. This is because the idle time for the first job in the sequence is not defined as the machines in front of it are empty. Fixing this to zero would have the effect of ensuring that the first job always has a low probability of being moved. As it is unlikely that the first job in the initial solution's sequence will be the correct choice, and so should be moved eventually, it is assigned the amount of idle time given above.

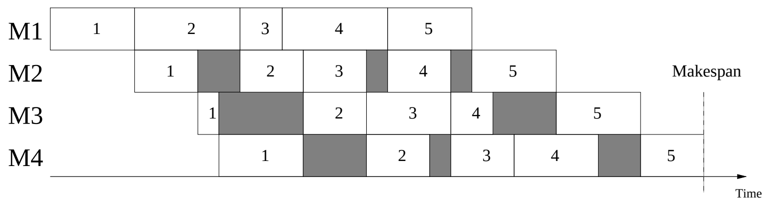

Fig. 5. An Example Gantt Chart for a 5x4 Flowshop

How can this information be used to predict which moves are most likely to lead to improvements in makespan? The answer is that first, having jobs waiting to be processed is not desirable as they represent delays; second, it is reasonable to suppose that jobs that are left waiting for a long time are in an unsuitable position in the sequence, and therefore should be moved.

The evaluation function was adapted to return the idle times associated with each job, as described above. When the neighbourhood operator was then applied, the position(s) of the move operator was decided on the basis of the idle time of the job at that position, by using one of either Tournament Selection [1] or Marriage Selection. Marriage Selection is a variant of tournament selection designed to soften its inherently high selection pressure. Possible jobs are picked at random until either one is found that is more attractive than the first, or a fixed number (the *Monte Carlo* size) of possible moves have been selected, in which case the job with the highest idle time associated with it is chosen [14].

5 Evaluating The Heuristic

The measure of performance used was the quality of solution obtained after a set number, N, of evaluations. Fifty runs were taken in each case, therefore summaries are given below and detailed results are available from the authors. Experiments were performed for the first of the Talliard test set's [16] instances of the following ($n \times m$) problems: 20x5, 20x10, 20x20, 50x5, 50x10, 50x20, 100x5, 100x10, 100x20, 200x10, and 200x20. The value of N used was solely dependent upon n and was set to 5000, 6750, 8000, and 9250 evaluations respectively. This is a roughly $ln(n)$ relationship, justified from empirical results from [12].

5.1 Stochastic Hillclimbing Results

Problem	Undirected	Heuristic + Random	Heuristic + Heuristic
20x5	1296.26 (3.63)	**1291.68 (7.52)** [T9]	**1294.58 (5.68)** [M4]
20x10	1633.32 (21.73)	1629.32 (21.15) [M5]	*1642.66 (27.37)* [T2]
20x20	2373.16 (26.94)	2367.80 (29.29) [M3]	2372.52 (23.68) [T2]
50x5	2742.82 (9.14)	**2728.90 (4.10)** [T9]	**2732.56 (6.62)** [T4]
50x10	3141.72 (24.76)	**3126.72 (19.98)** [M8]	3133.68 (25.85) [M3]
50x20	4049.34 (31.64)	**4037.62 (26.40)** [T3]	4046.38 (25.83) [T2]
100x5	5510.62 (12.91)	**5500.08 (11.75)** [T8]	**5495.80 (4.28)** [T8]
100x10	5927.38 (37.54)	**5893.12 (35.40)** [T9]	**5907.42 (33.17)** [M6]
100x20	6622.44 (44.06)	**6588.62 (35.34)** [M6]	**6605.48** (32.09) [M4]
200x10	11029.90 (35.02)	**10995.12 (33.04)** [T9]	**10992.40 (28.64)** [M8]
200x20	11826.10 (49.64)	**11764.14 (59.65)** [T8]	**11780.46 (54.43)** [M6]

Table 1. Stochastic Hillclimbing Results — Swap Neighbourhood

The results for stochastic hillclimbing are given in Tables 1 and 2. For all of the tables presented here, the mean and standard deviations of the makespan are given, with the selection type and pressure found to be most effective in square brackets.

Problem	Undirected	Heuristic + Random	Random + Heuristic	Heuristic + Heuristic
20x5	1293.60 (7.26)	1292.58 (7.81) [M4]	**1284.00 (10.32)** [T9]	**1288.52 (9.27)** [M3]
20x10	1618.92 (13.13)	1615.80 (10.62) [T2]	1615.12 (16.12) [T2]	1617.34 (16.64) [T2]
20x20	2358.88 (33.10)	2361.28 (25.90) [M3]	2352.30 (26.25) [T2]	2364.22 (27.15) [T2]
50x5	2734.46 (6.76)	**2731.98 (6.93)** [T8]	**2730.48 (5.15)** [M5]	**2730.92 (6.47)** [T4]
50x10	3121.10 (24.73)	3125.86 (19.89) [M8]	**3107.96 (21.59)** [T3]	3117.54 (18.77) [M3]
50x20	4021.62 (22.52)	4019.64 (23.58) [M5]	**4011.98 (25.19)** [M3]	4018.68 (21.74) [T2]
100x5	5506.56 (14.14)	**5498.70 (8.53)** [M5]	**5493.24 (0.65)** [T9]	**5493.98 (1.50)** [T8]
100x10	5884.80 (30.06)	*5895.82 (35.19)* [T2]	**5842.00 (20.49)** [T8]	**5874.08 (32.51)** [M6]
100x20	6581.78 (38.13)	6582.32 (36.89) [M6]	**6542.88 (36.09)** [T5]	**6556.70 (38.44)** [T2]
200x10	11020.20 (31.61)	**11007.94 (35.52)** [M5]	**10977.52 (38.20)** [T9]	**10982.40 (23.85)** [T6]
200x20	11797.80 (50.06)	**11770.14 (51.45)** [M3]	**11669.52 (48.67)** [T8]	**11715.88 (53.82)** [M5]

Table 2. Stochastic Hillclimbing Results — Shift Neighbourhood

The form of the information about the selection method is as follows: the capital letter represents the selection method used (M for marriage, T for tour-

Problem	Undirected	Heuristic + Random	Heuristic + Heuristic
20x5	1296.00 (4.30)	**1293.84 (6.54)** [T6]	1294.98 (5.29) [T3]
20x10	1619.72 (14.73)	1619.52 (17.07) [T6]	*1626.76 (15.30)* [T2]
20x20	2357.36 (24.21)	2355.22 (22.79) [T2]	2359.44 (21.21) [T2]
50x5	2741.14 (10.02)	**2730.84 (6.04)** [T9]	**2735.64 (7.14)** [T4]
50x10	3135.34 (21.37)	**3122.84 (22.93)** [T4]	3131.44 (20.66) [T2]
50x20	4044.52 (27.69)	**4027.76 (24.04)** [T5]	**4030.72 (25.49)** [T2]
100x5	5516.12 (13.28)	**5500.32 (10.63)** [T9]	**5498.40 (8.56)** [T9]
100x10	5939.50 (41.34)	**5880.32 (24.12)** [T8]	**5902.92 (24.74)** [M4]
100x20	6606.50 (31.98)	**6582.76 (31.96)** [T4]	**6586.84 (28.20)** [T2]
200x10	11035.90 (41.03)	**10992.02 (26.15)** [T9]	**10997.50 (22.37)** [T6]
200x20	11797.40 (52.15)	**11755.82 (57.91)** [T5]	**11753.02 (56.75)** [M3]

Table 3. Simulated Annealing Results — Swap Neighbourhood

Problem	Undirected	Heuristic + Random	Random + Heuristic	Heuristic + Heuristic
20x5	1295.36 (5.19)	1294.62 (5.57) [T6]	**1280.34 (5.23)** [T9]	**1291.34 (7.85)** [M2]
20x10	1609.08 (10.86)	1611.32 (10.22) [M9]	1607.86 (11.04) [M3]	1611.12 (10.72) [T2]
20x20	2340.68 (22.56)	2343.24 (20.03) [T2]	2338.96 (23.32) [T2]	2346.84 (25.16) [T2]
50x5	2737.70 (7.63)	**2734.34 (7.96)** [T2]	**2729.82 (5.63)** [T6]	**2731.90 (6.33)** [M5]
50x10	3116.48 (23.13)	3115.96 (21.62) [M3]	**3103.86 (22.42)** [M6]	3110.80 (20.40) [M3]
50x20	4007.50 (23.25)	4008.46 (24.52) [T5]	4003.22 (19.19) [M5]	4008.78 (18.85) [T2]
100x5	5506.14 (14.65)	**5499.00 (8.04)** [T9]	**5493.76 (0.97)** [T6]	**5494.12 (1.29)** [T4]
100x10	5888.60 (34.58)	5895.82 (31.93) [T2]	**5843.04 (27.54)** [M8]	**5869.80 (27.92)** [T2]
100x20	6562.66 (36.39)	6554.50 (27.33) [T2]	**6526.16 (37.05)** [T4]	**6536.58 (32.03)** [M9]
200x10	11023.30 (38.10)	11012.28 (37.97) [T9]	**10977.42 (31.08)** [T9]	**10984.04 (23.99)** [T9]
200x20	11750.30 (51.57)	**11721.22 (51.10)** [M4]	**11644.56 (47.62)** [T9]	**11674.68 (46.54)** [M8]

Table 4. Simulated Annealing Results — Shift Neighbourhood

nament), and the number gives the Monte Carlo size that was found to give the best performance over a number of test runs. Where the results are highlighted in **bold**, the result was significantly better than the undirected result. Results in *italics* indicate that the result in question was significantly worse. In all cases, significance of any comparisons were confirmed using a t-test.

In general, the shift neighbourhood (operator) gave better solutions than the swap neighbourhood, regardless of whether the neighbourhood was directed or not. Also, the heuristic was able to produce better solutions, though the performance appears to depend upon a number of factors.

The first of these factors was that the heuristic has a much more beneficial effect upon the problems with a large number of jobs — this is not particularly surprising as it could be argued that the smaller problems were not sufficiently difficult for the heuristic to make any impact. Also, the selection pressures that have to be used appear to be quite high — higher than would be used for an EA population for example. This indicates that the use of 'weaker' selection methods, such as the standard implementations of fitness-proportionate and rank-based selection should be avoided (this was confirmed in later experiments).

Finally, how to select the jobs to be moved depended upon the neighbourhood used. The best results for a swap operator were obtained by selecting the one of the jobs at random, whereas for a shift operator, selecting the first job at random when the second job was selected heuristically gave the best results.

Problem	Undirected	Heuristic + Random	Heuristic + Heuristic
20x5	1296.28 (3.08)	**1292.96 (6.67)** [T8]	1295.74 (3.77)
20x10	1620.30 (16.21)	1618.08 (11.96) [T3]	1625.48 (16.74) [T2]
20x20	2358.22 (22.21)	**2348.08 (20.54)** [M5]	2359.16 (24.07) [T2]
50x5	2739.02 (9.35)	**2731.42 (8.19)** [T9]	**2735.80 (8.09)** [T2]
50x10	3128.22 (28.89)	3120.84 (19.60) [M5]	3126.46 (19.96) [T2]
50x20	4031.50 (23.21)	4027.04 (24.48) [T3]	4029.38 (20.40) [T2]
100x5	5517.66 (15.28)	**5503.96 (13.60)** [M5]	**5498.46 (8.54)** [T9]
100x10	5921.44 (31.73)	**5881.18 (30.99)** [T9]	**5902.26 (28.12)** [M8]
100x20	6600.00 (34.02)	**6577.48 (34.64)** [T2]	**6583.22 (33.81)** [T2]
200x10	11027.60 (34.44)	**10990.90 (23.74)** [T8]	**11001.14 (25.87)** [T4]
200x20	11798.70 (44.20)	**11738.06 (48.00)** [T8]	**11745.58 (43.21)** [T2]

Table 5. Threshold Accepting Results — Swap Neighbourhood

Problem	Undirected	Heuristic + Random	Random + Heuristic	Heuristic + Heuristic
20x5	1295.30 (5.09)	1295.40 (4.63) [M6]	**1282.10 (5.18)** [T9]	**1293.02 (6.84)** [T3]
20x10	1604.64 (10.74)	1606.98 (11.66) [T2]	1604.82 (11.10) [T2]	*1609.74 (12.33)* [T2]
20x20	2341.72 (20.15)	2338.74 (20.17) [T2]	2335.14 (21.11) [T2]	2338.76 (20.10) [T2]
50x5	2735.90 (7.23)	2734.44 (9.17) [T5]	**2728.98 (4.56)** [T9]	**2731.68 (6.14)** [M5]
50x10	3115.56 (20.06)	3116.38 (21.57) [M9]	**3104.24 (20.43)** [T8]	3110.48 (19.16) [M3]
50x20	4007.74 (20.17)	4003.50 (20.29) [M2]	**3998.86 (19.19)** [T4]	**4000.38 (19.69)** [T7]
100x5	5507.98 (13.64)	**5499.20 (7.92)** [T9]	**5494.38 (2.75)** [T8]	**5494.68 (1.71)** [T6]
100x10	5889.48 (32.69)	5897.74 (26.39) [M8]	**5842.58 (21.70)** [T5]	**5872.64 (31.10)** [T3]
100x20	6552.14 (32.45)	6549.18 (34.28) [T2]	**6519.38 (27.50)** [T6]	**6530.64 (33.66)** [T2]
200x10	11017.40 (33.09)	11010.24 (30.76) [T8]	**10967.46 (20.10)** [T9]	**10985.76 (26.61)** [T9]
200x20	11736.00 (46.31)	**11703.50 (43.93)** [M3]	**11631.02 (36.63)** [T8]	**11661.32 (41.49)** [M4]

Table 6. Threshold Accepting Results — Shift Neighbourhood

5.2 Simulated Annealing Results

Simulated annealing was implemented, for all problems, with an initial temperature of 12, a final temperature of 0.05 with temperature changes every 100 evaluations according to a logarithmic cooling schedule. This was found to give satisfactory results in the formative experiments in this work. The results for simulated annealing are given in Tables 3 and 4. The results obtained were similar in nature to those obtained for SHC, though simulated annealing did improve solution quality somewhat.

5.3 Threshold Accepting Results

Threshold accepting was implemented, with an initial threshold of 9, and a final threshold of 0, with the threshold changing linearly every 100 evaluations — values that were found to give satisfactory results in formative experiments. The results obtained for threshold accepting were similar in nature to those obtained for SHC, though solution quality was improved (Tables 5 and 6).

5.4 Record-to-Record Travel Results

Record-to-record travel was implemented, for all of the test problems, with an threshold of 5 which was held constant, and which was found to give satisfactory results. The results for RTRT are given in Tables 7 and 8. Again, the results obtained were similar in nature to those obtained for SHC, though generally of higher quality.

Problem	Undirected	Heuristic + Random	Heuristic + Heuristic
20x5	1296.26 (3.63)	**1293.48 (6.57)** [T5]	**1294.72 (5.46)** [M3]
20x10	1619.02 (14.18)	1619.18 (14.58) [M5]	*1632.74 (21.15)* [M4]
20x20	2355.62 (22.71)	2358.20 (24.28) [T2]	*2373.82 (24.46)* [T2]
50x5	2737.66 (11.22)	**2733.04 (7.89)** [M9]	**2732.94 (6.74)** [T3]
50x10	3138.70 (22.01)	**3123.00 (19.41)** [M4]	3132.46 (22.27) [T2]
50x20	4044.50 (26.21)	**4030.00 (26.41)** [M5]	**4032.06 (16.51)** [T2]
100x5	5516.22 (15.10)	**5500.10 (10.93)** [T9]	**5497.38 (7.09)** [T8]
100x10	5917.56 (31.68)	**5883.72 (31.68)** [T8]	**5906.66 (27.92)** [M9]
100x20	6608.70 (32.57)	**6582.32 (33.18)** [M8]	**6589.02 (28.95)** [M8]
200x10	11033.50 (37.67)	**10993.44 (25.49)** [T9]	**10999.36 (29.19)** [M6]
200x20	11819.30 (37.84)	**11761.24 (50.38)** [M9]	**11764.22 (48.41)** [M4]

Table 7. Results for Record-to-Record Travel — Swap Neighbourhood

Problem	Undirected	Heuristic + Random	Random + Heuristic	Heuristic + Heuristic
20x5	1294.14 (6.18)	1294.84 (5.68) [M8]	**1281.32 (7.38)** [T8]	1293.42 (6.14) [M9]
20x10	1609.44 (9.36)	1609.58 (14.52) [T2]	1608.96 (12.39) [M4]	*1613.12 (10.58)* [T2]
20x20	2342.06 (23.12)	2349.18 (24.86) [M3]	2341.70 (22.66) [T2]	2349.62 (25.29) [T2]
50x5	2734.48 (8.22)	**2731.46 (7.26)** [T4]	**2729.50 (5.15)** [T9]	**2730.82 (6.02)** [T4]
50x10	3118.86 (22.65)	3114.46 (23.30) [T2]	**3106.14 (23.40)** [T3]	3113.06 (17.91) [T2]
50x20	4019.06 (22.79)	**4008.44 (18.45)** [T2]	**4005.30 (19.72)** [M5]	**4010.48 (21.58)** [M4]
100x5	5509.24 (15.00)	**5499.40 (8.58)** [T8]	**5493.82 (1.83)** [T9]	**5495.34 (4.16)** [T9]
100x10	5889.70 (33.16)	*5901.92 (31.62)* [T2]	**5838.00 (24.97)** [T9]	**5873.42 (24.88)** [M9]
100x20	6567.06 (37.40)	6568.18 (34.04) [M4]	**6533.82 (27.74)** [M5]	**6540.46 (28.62)** [M3]
200x10	11019.90 (39.49)	**11006.70 (28.39)** [T3]	**10977.66 (25.76)** [T8]	**10984.26 (23.36)** [T8]
200x20	11752.80 (55.88)	11741.34 (48.07) [M3]	**11647.04 (37.13)** [T9]	**11691.72 (41.39)** [M3]

Table 8. Results for Record-to-Record Travel — Shift Neighbourhood

5.5 Evolutionary Algorithm Results

Formative experiments were undertaken to find satisfactory settings for an EA across all of the test problems. This lead to the adoption of a steady-state EA with kill-worst replacement, a crossover probability of 0.3, and a mutation probability of 0.7. Selection of the first parent was performed using rank-based selection, and the second parent was selected at random. The results for the evolutionary algorithm are given in Tables 9 and 10. Unlike the other optimisation techniques, the evolutionary algorithm did not show significant improvements in performance when the neighbourhood was directed, except for the largest problems. Why should this be?

Examination of the results showed that the changes in makespan effected by the directed mutation operator for the EA were of roughly the same size as obtained for the other optimisers considered here. However the standard deviation of the EA results are much higher, which means that the improvements in

Problem	Undirected	Heuristic + Random	Heuristic + Heuristic
20x5	1268.76 (87.63)	1263.42 (90.56) [M5]	1267.66 (89.58) [M3]
20x10	1590.32 (80.20)	1588.32 (81.31) [M7]	1592.44 (80.14) [T2]
20x20	2284.18 (66.64)	2284.58 (73.89) [M6]	2287.56 (71.21) [T2]
50x5	2814.04 (129.00)	2805.04 (132.56) [T9]	2809.58 (129.22) [T2]
50x10	3174.40 (101.20)	3162.00 (98.97) [T8]	3162.66 (104.48) [M9]
50x20	4033.84 (88.94)	4021.42 (82.27) [M5]	4028.04 (86.97) [M4]
100x5	5401.38 (186.55)	5379.84 (190.89) [T9]	5388.70 (190.39) [T3]
100x10	5844.22 (148.34)	5817.38 (145.00) [T5]	5826.96 (145.02) [T3]
100x20	6828.34 (81.23)	6808.98 (92.08) [T3]	6818.50 (98.01) [M5]
200x10	11073.30 (208.54)	11024.76 (217.44) [T8]	11033.40 (216.75) [T3]
200x20	12243.10 (124.37)	12207.40 (131.87) [T3]	**12197.06 (128.21)** [T9]

Table 9. Evolutionary Algorithm Results — Swap Neighbourhood

Problem	Undirected	Heuristic + Random	Random + Heuristic	Heuristic + Heuristic
20x5	1263.06 (90.78)	1262.94 (91.54) [M4]	1260.76 (92.37) [T6]	1262.38 (90.82) [T2]
20x10	1576.00 (78.94)	1580.04 (79.90) [M4]	1575.80 (79.62) [T3]	1584.72 (77.28) [M3]
20x20	2271.08 (69.95)	2272.94 (74.36) [M4]	2270.36 (69.92) [T2]	2275.16 (73.77) [T7]
50x5	2809.12 (130.13)	2810.02 (129.63) [T4]	2798.82 (133.38) [T8]	2803.08 (131.50) [T2]
50x10	3165.16 (103.64)	3161.16 (104.29) [M4]	3145.20 (99.19) [T8]	3155.14 (101.41) [M6]
50x20	4024.14 (82.78)	4015.04 (86.37) [T2]	4009.60 (85.66) [T5]	4009.34 (86.06) [M8]
100x5	5390.96 (185.53)	5394.50 (186.07) [T2]	5372.00 (187.50) [T9]	5378.86 (190.34) [T3]
100x10	5831.74 (144.03)	5832.36 (149.05) [M5]	5800.58 (145.83) [T6]	5813.88 (148.60) [T3]
100x20	6836.50 (86.46)	6817.50 (98.39) [M5]	**6801.18 (90.38)** [T8]	**6804.46 (102.65)** [T2]
200x10	11072.90 (216.02)	11053.84 (219.50) [T8]	**11000.12 (209.69)** [T9]	11030.34 (223.48) [T9]
200x20	12267.60 (133.43)	12252.32 (115.24) [T3]	**12227.52 (113.99)** [T8]	**12222.56 (119.92)** [M9]

Table 10. Evolutionary Algorithm Results — Shift Neighbourhood

performance cannot be said to be statistically significant. As an aside, the much higher standard deviation obtained could be seen to be a reason for not using an EA for this problem: users would prefer a greater degree of certainty of what the final solution will be, and they may chose optimisers that provide this.

6 General Points

Although basic implementations of the optimisers have been used here, it would be interesting to compare optimiser performance. When the optimisers were compared few consistent patterns emerged, though the large variance associated with the evolutionary algorithm results may be a matter for concern. However, all of the implementations of the algorithms here possess considerable scope for improvement, so the choice of any one algorithm over another would be premature at this stage.

Turning our attention to representation, the shift neighbourhood consistently outperforms the swap neighbourhood. Why should this be so? A uninformative answer would be that the shift neighbourhood creates a more correlated landscape, and as all of these techniques work by exploiting such correlations, and directing the neighbourhood has no effect on this, we can account for the observation. But, why should a shift-neighbourhood create a more correlated search space? If example Gantt charts are examined, it soon becomes apparent that there are blocks of jobs that are 'well-meshed' together (ie. have little are no idle time between then). It would appear important that these blocks should be disrupted as little as possible by the neighbourhood operator, or better still, manipulated explicitly. This is precisely what the shift-operator does. In fact, an examination of example Gantt charts showed that the way the jobs in a block mesh together is largely preserved.

In the context of directing the neighbourhood, it was found that selecting both jobs according to the heuristic was not as effective as selecting just one. This is presumably because the former method over-exploits the idle-time information — using idle-time to predict improving moves is only a rule-of-thumb, therefore it is prudent to 'hedge your bets' somewhat, hence the stochastic nature of these algorithms.

The results obtained using the shift neighbourhood indicate that selecting the second job (that which defines the block to be shifted) is more important that selecting the first job (which defines which job is moved out of the way of the shifted block). This is probably connected to the reason why the shift operator is the operator of choice. If the movement of blocks is what is important, then how the block as a whole interacts with the other jobs will be important. As the idle time of the second selected job effectively measures how one end of the block 'fits in' with the rest of the schedule, it can be seen why it results in an effective measure for how blocks should be shifted.

7 Conclusions

On the basis of the results obtained here, the belief that moving jobs that have high idle times would lead to improved solutions appears to be valid. In addition, the method by which this belief is implemented does appear to have an effect, along with the choice of neighbourhood operator.

In addition, and possibly more importantly, the choice of neighbourhood and the directing of the neighbourhood were consistent across optimisation algorithms — a fact that, if noted, is rarely emphasised. The authors would suggest that this is worth noting: experience in mapping domain knowledge to these optimisers is often transferable which allows the user to more easily decide how domain knowledge should be exploited. Thus options can be explored before deciding to use a more complicated optimisation technique such as an evolutionary algorithm which introduces many more choices and parameters that require tuning.

Finally, it is worth mentioning that this idea can be transferred to other problems, such as job-shop scheduling. Also, beliefs can easily be formulated to deal with other optimisation criteria, for example, tardiness. In fact the operations research literature appeared, in the course of writing this paper, to be a good source of such heuristics and users of evolutionary optimisers would do well to turn their attention there.

8 Acknowledgements

We would like to express our gratitude to the Engineering and Physical Sciences Research Council (EPSRC) for their support of Andrew Tuson via a research studentship (95306458). In addition, we would like to thank the reviewers for the references they provided and Chris Gathercole for final proofreading.

References

1. A. Brindle. *Genetic Algorithms for Function Optimization.* PhD thesis, University of Alberta, 1981.
2. G. Dueck. New Optimisation Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel. Technical report, IBM Germany, Heidelburg Scientific Center, 1990.
3. G. Dueck and T. Scheuer. Threshold Accepting: A General Purpose Optimisation Algorithm Superior to Simulated Annealing. *Journal of Computation Physics*, 90:161–175, 1990.
4. C. Hjorring. *The Vehicle Routing Problem and Local Search Metaheuristics.* PhD thesis, University of Queensland, Australia, 1995.
5. J.C. Ho and Y.-H. Chang. A new heuristic for the n-job, M-machine flow-shop problem. *European Journal of Operational Research*, 52:194–202, 1991.
6. J.H. Holland. *Adaptation in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press, 1975.
7. A. H. G. Rinnooy Kan. *Machine Sequencing Problems: Classification, complexity and computations.* Martinus Nijhoff, The Hague, 1976.
8. S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.
9. S. Minton, A. Phillips, A. Johnston, and P. Laird. Solving Large Scale CSP and Scheduling Problems with a Heuristic Repair Method. In *Proceedings of AAAI-90*, 1990.
10. G. F. Mott. Optimising Flowshop Scheduling Through Adaptive Genetic Algorithms. Chemistry Part II Thesis, Oxford University, 1990.
11. E. Nowicki and C. Smutnicki. A fast tabu search algorithm for the permutation flow-shop problem. *European Journal of Operational Research*, 91:160–175, 1996.
12. I. H. Osman and C. N. Potts. Simulated annealing for permutation flow-shop scheduling. *OMEGA*, 17:551–557, 1989.
13. C. Rajendran and D. Chaudhuri. An efficient heuristic approach to the scheduling of jobs in a flowshop. *European Journal of Operational Research*, 61:318–325, 1991.
14. Peter Ross. Personal communication, 1996.
15. Peter Ross, Dave Corne, and Hsiao-Lan Fang. Improving evolutionary timetabling with delta evaluation and directed mutation. In Y. Davidor, H-P. Schwefel, and R. Manner, editors, *Parallel Problem-solving from Nature - PPSN III*, LNCS, pages 566–565. Springer-Verlag, 1994.
16. E. Taillard. Benchmarks for basic scheduling problems. *European Journal of operations research*, 64:278–285, 1993.
17. T. Yamada and R. Nakano. Scheduling by Genetic Local Search with Multi-Step Crossover. In *The Fourth International Conference on Parallel Problem Solving From Nature (PPSN IV)*, 1996.

18. M. Zweben and E. Davis. Learning to Improve Iterative Repair Scheduling. Technical report, NASA Ames Research Centre, Report FIA-92-14, 1992.
19. M. Zweben, E. Davis, B. Daun, and M. Deale. Scheduling and Rescheduling with Iterative Repair. Technical report, NASA Ames Research Centre, Report FIA-92-16, 1992.