A Heuristic Combination Method for Solving Job-Shop Scheduling Problems

Emma Hart, Peter Ross

Department of Artificial Intelligence, University of Edinburgh,
Edinburgh EH1 2QL, Scotland

Abstract. This paper describes a heuristic combination based genetic algorithm, (GA), for tackling dynamic job-shop scheduling problems. Our approach is novel in that the genome encodes a choice of algorithm to be used to produce a set of schedulable operations, alongside a choice of heuristic which is used to choose an operation from the resulting set. We test the approach on 12 instances of dynamic problems, using 4 different objectives to judge schedule quality. We find that our approach outperforms other heuristic combination methods, and also performs well compared to the most recently published results on a number of benchmark problems.

1 Introduction

Job-Shop scheduling problems are known to be NP-hard, and cannot be solved in polynomial time. Many approximate and heuristic-based methods have been proposed to attempt to find near-optimal solutions to large problems in realistic time-scales, a large number of which include the use of genetic algorithms. A wide variety of GA techniques have been proposed, using many different representations and operators, and relating to a broad spectrum of problems, e.g. [2].

A common problem encountered in using a GA for scheduling problems is how to represent a problem on a chromosome. Early work in the field yielded two 'extreme' approaches to representing schedules. Nakano and Yamada, [9], used a binary representation to encode schedules for benchmark job-shop problems. A complex effort was required to design such an encoding, and its use in a GA context needed specialised repair operators to retain the ability to decode chromosomes as feasible schedules. The other 'extreme', for example [1], is a 'direct encoding' of a schedule, in which an encoded schedule was a direct representation of the schedule itself, with data structures and attributes designed to simply mimic the real schedule. Hence no decoding of a chromosome was necessary, but the genetic operators needed to use much domain and constraint knowledge to appropriately mutate schedules, and to maintain feasible schedules. A somewhat in-between approach uses an 'indirect' encoding in which a schedule building algorithm is combined with genetic search through the space of potential inputs to the schedule builder, for example [5].

An interesting subclass of representations are those loosely defined as 'Heuristic Combination Methods', (HCM), varieties of which have been reported by

Dorndorf, [3], Fang, [5], and Norenkov, [10]. These methods all use an implicit representation of a schedule in which each gene in the chromosome represents a heuristic to be used at each step of generating a schedule. We present a new version of an HCM, and compare its performance to several other representations on a number of problems. In particular, we concentrate on dynamic job shop problems, as these have most relevance to the nature of most real-world problems. Several different schedule quality measures are considered for each problem, in order to evaluate the robustness of the algorithm.

2 Schedule Generation

The general job shop problem assumes J jobs have to be processed on M machines, in a pre-defined order. Each operation of job j on machine m is denoted by (j_i, m_i), and has a processing time of p_{jm}. The total processing time of a job is given by P_j. In a dynamic job-shop problem, each job has a release date r_j, and a due-date d_j by which time it must complete. The importance of each job is given by a weight w_j. A machine can only process one machine at a time, and preemption of any operation on a machine is not allowed. Feasible schedules fall into four classes; inadmissible, semi-active, active and non-delay. Figure 1,[4], illustrates the relationship between each type of schedule. There are an infinite number of *inadmissible* schedules, which contain excess idle time, and therefore are of no interest. *Semi-Active* schedules contain no excess idle time. *Active* schedules contain no idle time, and furthermore, have no operations which can be completed earlier without delaying other operations. The optimal schedule is guaranteed to fall within the set of active schedules. *Non-Delay* schedules are a subset of active schedules, in which operations are placed into the schedule such that no machine is ever kept idle if some operation is able to be processed on it.

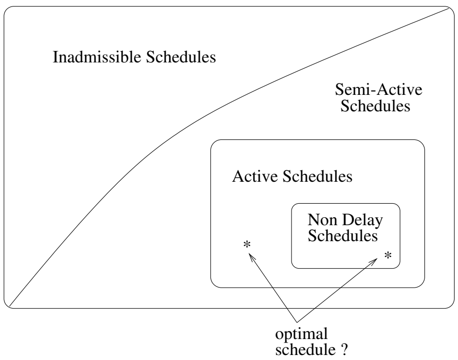

Fig. 1. Types of feasible schedule

Active schedules can be generated using the Giffler and Thompson algorithm

[6], (G&T), which is given in figure 2. Non-delay schedules are generated by a modification of this algorithm shown in figure 3, in which the operation chosen to be scheduled is always the one that can start earliest. Both methods result in a conflict set of operations, G, from which an operation must be chosen to be scheduled.

Dorndorf and Pesch [3] proposed a 'priority-rule based' genetic algorithm, incorporating the G&T algorithm. This used a chromosome $(p_1, p_2 \ldots p_{j*m})$, where gene p_i encoded a dispatch rule to be used to resolve the conflicts produced as the i_{th} iteration of the G&T algorithm. They tested this algorithm on a number of static job-shop benchmark problems, using makespan as the objective, obtaining results which have since been superseded by other methods, for example see [11]. Our approach extends this by noticing from figure 1 that as non-delay schedules are a subset of all active schedules, in some cases it may be sufficient to search the space of *non-delay* schedules rather than all *active* schedules. As we do not know *a priori* what type of schedule is optimal, as well as encoding a heuristic rule, our approach also encodes on the chromosome the *methodology*, (i.e G&T or Non-Delay), to use to construct the conflict set each time an operation needs to be scheduled.

1. Calculate the set C of all operations that can be scheduled next

2. Calculate the completion time of all operations in C, and let m^* equal the machine on which the minimum completion time t is achieved.

3. Let G denote the conflict set of operations on machine m^* - this is the set of operations in C which take place on m^*, and whose start time is less than t.

4. Select an operation from G to schedule

5. Delete the chosen operation from C and return to step 1.

Fig. 2. Giffler and Thompson Algorithm

Thus, each gene in the chromosome encodes a pair $(Method, Heuristic)$, which denotes the method that should be used to calculate the conflicting set of schedulable operations at iteration t, and the heuristic to be used to select an operation from the set. The representation guarantees a feasible solution, and straightforward recombination operators can be used which always produce feasible offspring. We refer to this method as *HGA – heuristically-guided GA*.

1. Calculate the set C of all operations that can be scheduled next

2. Calculate the starting time of each operation in C and let G equal the subset of operations that can start earliest

3. Select an operation from G to schedule

4. Delete the chosen operation from C and return to step 1.

Fig. 3. Non Delay Algorithm

3 Test Problems and Heuristics

We test our approach on 12 different dynamic job-shop scheduling problems taken from Morton & Pentico,[8], using four normalised weighted objective functions; Weighted Earliness plus Tardiness (ETwt), Weighted Flowtime (Fwt), Weighted Tardiness (Twt) and Weighted Lateness (Lwt). If we define the completion time of job j as C_j, then the lateness, L_j, of a job j can be defined as $C_j - d_j$, the earliness, E_j, as $max(-L_j, 0)$, and the tardiness, T_j, as $max(L_j, 0)$. The definitions of the objective functions are given in table 1. 12 heuristics are defined to select between operations in the conflict set. These are shown in table 2. With the exception of the RND heuristic, all of these heuristics take account of the weighting attached to each job. RND is included to ensure that every job in the conflict set can be chosen by at least one heuristic, as it is not possible to guarantee this otherwise.

Objective Function	Objective	Normalised Definition
ETwt	Minimize	$\left(\sum_j w_j(E_j + T_j)\right) / \left(\sum_j w_j P_j\right)$
Fwt	Minimize	$\left(\sum_j w_j(C_j - r_j)\right) / \left(\sum_j w_j P_j\right)$
Twt	Minimize	$\left(\sum_j w_j T_j\right) / \left(\sum_j w_j P_j\right)$
Lwt	Minimize	$\left(\sum_j w_j L_j\right) / \left(\sum_j w_j P_j\right)$

Table 1. Objective Function Definitions

4 GA Parameters

We use a parallel GA, with 5 sub-populations of size 50 arranged in a ring, with migration of one chromosome from one population to the next occurring every

Rule	Description
WSPT	Weighted shortest processing time
WLWKR	Weighted Least Work Remaining
WTWORK	Weighted Total Work
EGD	Earliest Global Due Date
EOD	Earliest Operational Due Date
EMOD	Earliest Modified Operational Due Date
MST	Modified Slack Time
SOP	Slack per Operation
POPNR	Lowest ratio of processing time of imminent operation to weighted value of remaining operations
PSOP	Weighted smallest sum of (next processing time + SOP)
PWKR	Weighted smallest ratio of processing time to work remaining
RND	Choose a Random operation from those operations that can't be chosen by another heuristic

Table 2. Heuristics Used for Dynamic Job-Shop Problems

5 generations. The length of a chromosome is equal to the number of operations to be scheduled. Uniform crossover is used, and crossover always takes place between gene pairs, so that an (M, H) schema is never destroyed by crossover. A mutation operator mutates each heuristic in the genome to another randomly chosen heuristic with probability $p = 0.01$. For each heuristic mutated, the corresponding method allele is mutated with probability 0.5. We use a generational reproduction strategy, with rank selection. All experiments are run for 1000 generations, to allow a fair comparison with the experiments of [7] and [4].

5 Experiments

The results for the dynamic problems are compared to those obtained using Priority Rules, to those of Fang, [4], and those most recently reported by Lin *et al*, [7] using a parallel GA they refer to as PGA. In each case we report the best result found in 10 repeated experiments, (as did both [7] and [4]). We also compare the results using HGA to experiments using the heuristic representation, but in which the choice of algorithm was restricted to either completely G&T or completely Non-Delay, to gauge the effectiveness of evolving choice of algorithm alongside the choice of heuristic. These experiments are referred to as HGA(G&T) and HGA(ND). respectively.

The final series of experiments investigated the effect of not including the RND heuristic in the set of alleles, HGA(NR). Inclusion of this heuristic has the undesirable effect that repeated evaluations of the same chromosome may result in different schedules, depending on the interpretation of the RND heuristic, and hence a chromosome that was fit in one generation may suddenly become unfit in the next. This obviously has undesirable consequences for the stability of a GA population, as good solutions from one generation may be eliminated entirely at the next generation.

	Size	Fang	Pri.	PGA	HGA	HGA (ND)	HGA(G&T)	HGA (NR)
jb1	10x3	0.475	0.529	0.474	0.474	0.527	0.474	0.474
jb2	10x3	0.758	0.758	0.499	0.753	0.905	0.753	0.753
jb4	10x5	0.620	0.622	0.621	0.619	0.620	0.619	0.619
jb9	15x3	0.384	0.395	0.369	0.370	0.379	0.381	0.374
jb11	15x5	0.263	0.415	0.262	0.271	0.403	0.271	0.271
jb12	15x5	0.247	0.494	0.246	0.247	0.488	0.247	0.247
ljb1	30x3	0.322	0.654	0.279	0.280	0.653	0.280	0.280
ljb2	30x3	0.632	0.868	0.601	0.602	0.751	0.612	0.598
ljb7	50x5	0.374	0.515	0.254	0.284	0.449	0.287	0.269
ljb9	50x5	1.178	0.935	0.739	0.832	0.854	0.967	0.797
ljb10	50x8	0.621	0.882	0.598	0.566	0.691	0.548	0.567
ljb12	50x8	0.607	0.667	0.461	0.522	0.469	0.473	0.466

Table 3. Weighted Earliest + Tardiness (ETwt)

	Size	Fang	Pri.	PGA	HGA	HGA (ND)	HGA (G&T)	HGA (NR)
jb1	10x3	1.237	1.231	1.231	1.236	1.236	1.236	1.236
jb2	10x3	1.778	1.772	1.768	1.766	1.771	1.766	1.766
jb4	10x5	1.109	1.111	1.108	1.107	1.112	1.107	1.107
jb9	15x3	1.768	1.947	1.754	1.754	1.759	1.760	1.754
jb11	15x5	1.794	1.795	1.706	1.706	1.716	1.834	1.706
jb12	15x5	1.259	1.257	1.256	1.256	1.258	1.256	1.256
ljb1	30x3	1.431	1.494	1.391	1.389	1.487	1.389	1.389
ljb2	30x3	1.826	1.924	1.777	1.777	1.804	1.837	1.782
ljb7	50x5	1.669	1.692	1.557	1.548	1.563	1.709	1.542
ljb9	50x5	2.659	2.490	2.324	2.349	2.394	2.626	2.337
ljb10	50x8	1.728	1.776	1.697	1.673	1.680	1.761	1.675
ljb12	50x8	2.138	2.207	2.080	2.070	2.058	2.237	2.068

Table 4. Weighted FlowTime (Fwt)

6 Results

The results are shown in tables 3,4,5, and 6. *HGA* outperforms the GA reported by Fang in 40 out of 48 cases, produces equal results for 5 cases, and is only

	Size	Fang	Pri.	PGA	HGA	HGA (ND)	HGA (G&T))	HGA (NR)
jb1	10x3	0.164	0.178	0.162	0.163	0.180	0.163	0.163
jb2	10x3	0.087	0.086	0.086	0.086	0.086	0.086	0.086
jb4	10x5	0.556	0.560	0.559	0.556	0.557	0.556	0.556
jb9	15x3	0.177	1.185	0.169	0.170	0.170	0.181	0.170
jb11	15x5	0.000	0.000	0.000	0.000	0.000	0.000	0.000
jb12	15x5	0.139	0.218	0.139	0.139	0.219	0.139	0.139
ljb1	30x3	0.215	0.276	0.190	0.194	0.279	0.192	0.194
ljb2	30x3	0.459	0.460	0.395	0.407	0.419	0.437	0.407
ljb7	50x5	0.110	0.109	0.060	0.056	0.069	0.091	0.055
ljb9	50x5	0.982	0.796	0.651	0.652	0.674	0.917	0.630
ljb10	50x8	0.455	0.479	0.438	0.418	0.442	0.438	0.418
ljb12	50x8	0.478	0.489	0.399	0.392	0.397	0.424	0.399

Table 5. Weighted Tardiness (Twt)

	Size	Fang	Pri.	PGA	HGA	HGA (ND)	HGA (G&T)	HGA (NR)
jb1	10x3	-0.168	-0.173	-0.173	-0.167	-0.167	-0.167	-0.167
jb2	10x3	-0.824	0.812	-0.839	-0.819	-0.814	-0.816	-0.819
jb4	10x5	0.490	0.497	0.493	0.489	0.494	0.489	0.489
jb9	15x3	-0.073	-0.047	-0.078	-0.079	-0.073	-0.072	-0.079
jb11	15x5	-0.607	-0.607	-0.751	-0.643	-0.702	-0.580	-0.636
jb12	15x5	-0.102	-0.082	-0.103	-0.102	-0.101	-0.102	-0.102
ljb1	30x3	-0.195	-0.112	-0.214	-0.224	-0.126	-0.224	-0.224
ljb2	30x3	-0.043	0.013	-0.078	-0.078	-0.051	-0.022	-0.051
ljb7	50x5	-0.414	-0.411	-0.507	-0.514	-0.501	-0.399	-0.523
ljb9	50x5	0.702	0.622	0.354	0.389	0.423	0.595	0.388
ljb10	50x8	-0.096	-0.038	-0.108	-0.139	-0.136	-0.041	-0.134
ljb12	50x8	0.256	0.256	0.113	0.158	0.121	0.310	0.136

Table 6. Weighted Lateness (Lwt)

beaten in 3 cases. Comparison to the results obtained using Priority Rules shows that HGA outperforms them in 44 out of 48 cases, equals the results in 2 cases, and is beaten in 2 cases. The performance of HGA is less robust when compared to those results recently published by Lin. However, it outperforms Lin's PGA in 18 cases, equals the results in a further 9 cases, and it is beaten in the remaining 21 cases. HGA(NR) beats Lin's results in an additional 2 cases. The objective for which HGA produces the poorest results is ETwt, and the best results are found for objective Fwt.

For all objectives, evolving the choice of scheduling method is beneficial. Comparison of HGA to G&T only shows that in only 3 cases does the evolution of method hinder the search and decrease performance. For the remaining cases, HGA produces better results than $HGA(G\&T)$ in 25 cases, and equivalent results for the other 20 cases. Comparison to Non-Delay shows 4 cases where better results are achieved using Non-Delay only. HGA produces better results than $HGA(ND)$ on 40 cases, and equivalent results on the remaining 4 cases.

The decision as to whether or not to include the RND heuristic is less clear. Comparison of the results obtained by HGA and HGA(NR) shows that in just over half the cases (28) no difference between results is observed. From the remaining 20 cases, better results are obtained in 9 cases by including the RND allele, and worse results on 11 cases. Hence, although not including the RND allele may prevent some operations from being chosen at some points in the schedule, this appears to slightly outweigh the possible disadvantages of including it, i.e. that the population may be unstable from one generation to the next.

7 Analysis and Discussion

Both G&T and Non-Delay algorithms result in a conflict set of operations at each iteration, of size ≥ 1. If the conflict set has only one operation, then any of the available h heuristics will obviously select this operation. If we denote the number of conflict sets of size > 1 by d, then the size of the search space is equal to d^h.

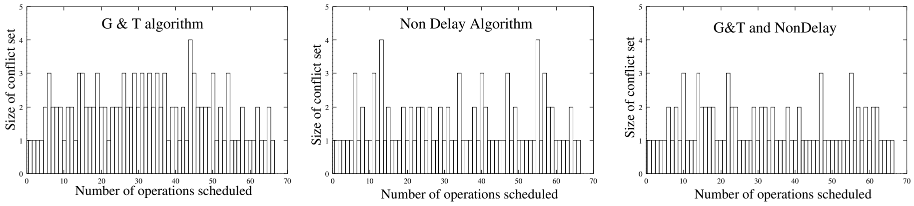

Fig. 4. Size of conflict set vs number of operations

We investigate the variation on conflict set size as operations are scheduled using the best chromosome evolved by HGA for problem $ljb1$, objective Fwt. The result is shown in figure 4, which also contrasts the effect of setting the *method* genes to denote G&T only and to Non-Delay only, leaving the *heuristic* genes fixed.

Use of G&T only leads to a high value of d, and hence a large search space. On the other hand, Non-Delay schedule generation results in a low value of d, but may exclude the optimal schedule from consideration. By evolving the algorithmic choice, we reach a compromise between the two schemes. These diagrams also give us a clue as to why HGA is able to perform successfully — in each chromosome there are a large number of redundant *heuristic* alleles, which can effectively have any value. Thus many *different* chromosomes may evaluate to the *same* schedule.

It also seems reasonably intuitive that the crucial stages in the scheduling process lie in constructing the early part of the schedule, where the knock-on effect on making some decision can have serious consequences on the remainder of the schedule. This implies that some of the decisions made in the latter part of the schedule are of little importance. This points to a possible alternative method of reducing the search space; that is concentrate on evolving some crucial partial schedule, then generate the remainder by other means. Consider the following experiment; take the best chromosome output by HGA for problem $ljb1$, and replace the final i heuristic genes with heuristics chosen at random. Repeat the experiment, with i ranging from 1 to l, where l is the length of the chromosome. For each value of i, 250 chromosomes were generated and evaluated. Figure 5 shows the best objective found at each value, with the best objective found by Lin, Fang, and Priority rules also shown for comparison. The diagram shows that placement of the first 34 of the total 67 operations in the schedule is critical, after which random generation of the remainder produces satisfactory results.

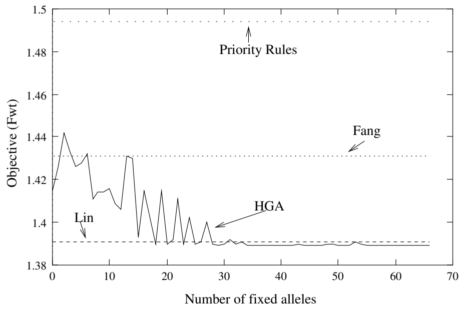

Fig. 5. Best Fwt objective obtained for ljb1 vs number of alleles fixed

8 Conclusion

This paper presented a new GA which evolves a combination of scheduling algorithm and heuristic choice to be used at each stage of the scheduling process. In particular, evolving the choice of scheduling algorithm is shown to be highly beneficial. The representation is straightforward, and can be used with standard recombination operators, which always produce feasible solutions. The results are promising across a range of problems and objectives when compared to those most recently published. The GA could perhaps be tuned further with respect to individual objectives by using a specialised set of heuristics for the objective in question, rather than the general set used here.

Furthermore, analysis of the behaviour of the algorithm suggests several ways in which it may be improved. The diagrams presented in section 7 suggest that it may be feasible to concentrate on evolving a *partial* schedule, with an obvious reduction in search space size, and then rapidly produce the remainder of the schedule via another more efficient means, perhaps by hybridisation with a hill-climber. Similarly, more efficient search maybe produced by directing mutation and crossover operators to those areas of the chromosome where the conflict set produced as a result of applying the scheduling algorithm is largest. However, the results found so far using "off-the-shelf" operators are extremely encouraging.

Acknowledgements

Emma Hart is supported by EPSRC grant GR/L22232.

References

1. R. Bruns. Direct chromosome representation and advanced genetic algorithms for production scheduling. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, page 352. San Mateo: Morgan Kaufmann, February 1993.
2. R. Bruns. Scheduling. In T. Bäck, D.B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, release 97/1, chapter Part F: Applications of Evolutionary Computing, pages F1.5:1–9. IOP Publishing Ltd and Oxford University Press, 1997.
3. U. Dorndorf and E. Pesch. Evolution based learning in a job shop scheduling environment. *Computers and Operations Research*, 22(1):25–40, 1995.
4. H-L. Fang. *Genetic Algorithms in Timetabling and Scheduling*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1994.
5. H-L. Fang, P. Ross, and D. Corne. A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 375–382. San Mateo: Morgan Kaufmann, 1993.
6. B. Giffler and G.L. Thompson. Algorithm for solving production scheduling problems. *Operations Research*, 8(4):487–503, 1960.
7. S-C. Lin, E.D. Goodman, and W.F. Punch. A genetic algorithm approach to dynamic job-shop scheduling problems. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 481–489. Morgan-Kaufmann, 1997.
8. T.E. Morton and D.W. Pentico. *Heuristic Scheduling Systems*. John Wiley, 1993.
9. R. Nakano and T. Yamada. Conventional genetic algorithms for job shop problems. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 474–479. San Mateo: Morgan Kaufmann, 1991.
10. I.P. Norenkov and E.D. Goodman. Solving scheduling problems via evolutionary methods for rule sequence optimization. Second World Conference on Soft Computing, 1997.
11. R.J.M. Vaessens, E.H.L. Aarts, and J.K. Lenstra. Job shop scheduling by local search. *INFORMS Journal of Computing*, 8:302–317, 1996.