A Promising Hybrid GA/Heuristic Approach for Open Shop Scheduling Problems

Hsiao-Lan Fang, Peter Ross,
and Dave Corne

DAI Research Paper No. 699

In Proceedings of the 11th European Conference on Artificial Intelligence, John Wiley and Sons, 1994, pages 590–594.

A Promising Hybrid GA/Heuristic Approach for Open-Shop Scheduling Problems

Hsiao-Lan Fang[1] and **Peter Ross**[1] and **Dave Corne**[2]

Abstract. Many problems in industry are a form of open-shop scheduling problem (OSSP). We describe a hybrid approach to this problem which combines a Genetic Algorithm (GA) with simple heuristic schedule building rules. Excellent performance is found on some benchmark OSS problems, including improvements on previous best-known results. We describe how our approach can be simply amended to deal with the more complex style of open shop scheduling problems which occur in industry, and discuss issues relating to further improvement of performance and integration of the approach into industrial job shop environments.

1 INTRODUCTION

The Open-Shop Scheduling Problem (OSSP) is a complex and common industrial problem [6]. OSSPs arise in an environment where there is a collection of operations to perform on one or more machines. Efficient production and manufacturing demands effective methods to optimise various aspects of a schedule, usually focussing on the total time taken to process all of the operations. We present a hybrid GA/heuristic approach which performs very successfully in comparison with previous results on some simple benchmark OSSPs [12]. In two cases (for which global optima had not already been found), our results improve on a previously best known result produced by tabu search [12]. Our approach is flexible and easy to use in terms of development time, and also exhibits several areas for future improvement.

We concentrate on three chromosome representation strategies. One is a straightforward extension to the OSSP of a strategy used for the job-shop scheduling problem (JSSP) in earlier work [5], which does not involve any hybridisation. The other two strategies incorporate simple means of hybridising the GA with heuristic rules. One of these methods seems more powerful and robust than the other two.

We also note that the benchmark problems used can be freely obtained for comparative research, and describe how our approach can be extended to address more complex open shop scheduling problems. We know of no GA-based efforts except ours on the OSSP with which to compare, so we present results in order to show the potential for a GA approach to open-shop scheduling, and invite fellow researchers to experiment with the same problems.

[1] University of Edinburgh, Department of Artificial Intelligence, 80 South Bridge, Edinburgh, EH1 1FN, UK

[2] University of Edinburgh, Department of Artificial Intelligence, 5 Forrest Hill, Edinburgh, EH1 2QL, UK

1.1 Overview

Section 2 describes the OSSP in detail, and our GA approach is described in section 3. Experiments and results on benchmark problems are then presented in section 4. Section 5 discusses these results, advances various issues concerning performance improvement, and notes how the approach may be extended to cope with more complex OSSPs.

2 OPEN-SHOP SCHEDULING PROBLEMS

A commonly used simplification of the OSSP is to specify that each given operation can only be processed on a given specified machine. In reality, an operation can often be processed in a number of alternative ways, any of which may involve more than one machine. There may also be due dates and machine setup times to consider. In the following however we will concentrate on a simplified form of the general problem; this is done mainly because the benchmark problems on which we test the performance of our GA approach are thus simplified. We will later discuss simple amendments to our approach which promise to successfully cope with the more general problem.

An OSSP involves a collection of m machines and a collection of j jobs; each job comprises a collection of operations (sometimes called tasks). An operation is an ordered pair (a, b), in which a is the machine on which the operation must be performed, and b is the time it will take to process this operation on machine a. A feasible OSSP schedule assigns a start time to each operation, satisfying the constraint that a machine can only process one operation at a time, and that two or more operations from the same job cannot be processed at the same time. The main objective is usually to generate a schedule with a makespan as short as possible; the makespan is simply the total elapsed time in the schedule. More complex objectives often arise in practice, where due dates and machine set up times must also be taken into account, for example.

The common illustration of this kind of problem is that of an automotive repair shop [6]. In such a shop, a typical job might involve the operations 'spray-paint', and 'change-tyres' to be performed on the same vehicle. These operations cannot usually be performed concurrently (especially if the stations at which these operations are performed are in different places, for instance), but can be performed in any order. Also it is usually true that different stations (ie: 'machines'

ECAI 94. *11th European Conference on Artificial Intelligence* Edited by A. Cohn
Published in 1994 by John Wiley & Sons, Ltd.

) can concurrently process operations from different jobs (eg: involving different vehicles). If the operations in a job must be performed in some fixed order, then this becomes a 'Job-Shop Scheduling Problem' (JSSP).

Certain benchmark OSSPs have been used for comparative research. In these, each job comprises precisely one operation for each machine. These benchmarks are hence completely defined by an ordered collection of m processing times for each job. For example, table 1 shows a 5×5 (ie: 5 jobs and 5 machines) benchmark problem, taken from [9].

Table 1. A 5x5 benchmark OSSP

Machines:	1	2	3	4	5
Job 1:	64	66	31	85	44
Job 2:	7	69	68	14	18
Job 3:	74	70	60	1	90
Job 4:	54	45	98	76	13
Job 5:	80	45	10	15	91

In the above example, operation 1 of job 1 must go to machine 4 for 85 units of processing time, operation 2 of job 1 must go to machine 1 for 64 units of processing time, and so on, with no restrictions on the order in which the tasks for any job are to be processed. The problem is to generate a valid schedule with minimal makespan. Figure 1 shows a minimum-makespan (300) schedule for the benchmark in table 1.

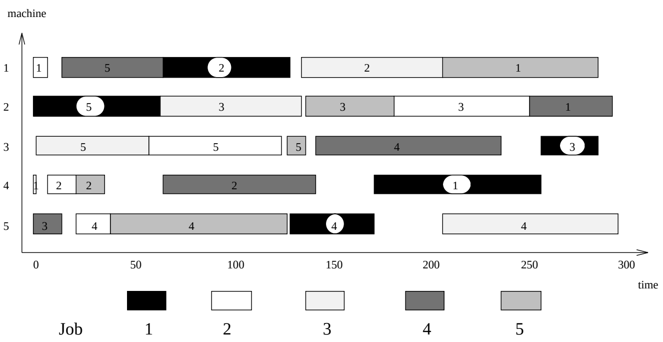

Figure 1. Minimal-makespan schedule for a 5x5 OSSP benchmark

3 A GA/HEURISTIC APPROACH TO THE OSSP

A common general technique for hybridising a GA with a heuristic search or heuristic rule based method is to use the GA to search a space of *abstractions* of solutions, and employ a heuristic or some other method to convert the points delivered by the GA into candidate solutions. Such hybridisation is one way of avoiding the often highly complicated problem of representing a complete solution as a chromosome in a way that facilitates effective GA-based search; it is usually more easy to represent abstract regions of the solution space, and have these abstractions converted into (ie: interpreted as) solutions by some other technique.

This paper presents some simple examples of such hybrid GA/heuristic methods for the OSSP. Similar GA/heuristic hybridisation occurs variously in the GA literature. Eg, a recent discussion of hybrid GA/heuristic hybrids for bin-packing and related problems appears in [11]. In the following, we describe three simple strategies for using a GA to address an OSSP.

3.1 Basic chromosome representation

Each of the representations we discuss is based on the following basic technique. The genotype for a problem is a string of p genes, where p is the total number of operations involved, summed over each job. Each gene can take alleles in the range $\{1,2,\ldots,j\}$, where j is the largest job number. A chromosome provides instructions for building a schedule as follows: the string of genes $abc\cdots$ means: "choose an untackled operation from the the a-th uncompleted job, and place it in the earliest place where it will fit in the developing schedule, choose an untackled operation from the b-th uncompleted job and place it into the earliest place where it will fit in the developing schedule, ...", and so on. Building a schedule is accomplished by a schedule builder, which maintains a circular list of uncompleted jobs and a list of untackled operations for each such job. Thus the notion of "a-th uncompleted job" is taken modulo the length of the circular list to find the actual uncompleted job being referred to.

Evidently, this description is incomplete because of the word "choose" in the interpretation method. In this sense, each chromosome represents a region of the space of possible solutions. For example, the region of solutions which may be represented by the chromosome "1,2,1,..." is that in which the first operation scheduled comes from job 1, the second from job 2, the third from job 1, and so on. The way that a chromosome is interpreted as a single solution somewhere in this region can vary. In this paper we look at three ways of doing this.

3.2 Directly encoding the operation

This is the most straightforward method; we simply double the size of the chromosome by incorporating genes for the choice of operation in addition to those for choice of job. Hence, $abcd\cdots$ will now mean: "choose the a-th untackled operation from the the b-th uncompleted job, and place it in the earliest place where it will fit in the developing schedule, choose the c-th untackled operation from the d-th uncompleted job and place it into the earliest place where it will fit in the developing schedule, ...", and so on. We will refer to this method with the abbreviation JOB+OP.

3.3 Fixed heuristic choice

In this method, a fixed heuristic is decided upon beforehand, and used by the schedule builder to make the choice of operation at each step. Hence, if we use heuristic X to make this choice, then the interpretation of $abcd\cdots$ becomes : "use heuristic X to choose an operation from the a-th uncompleted job, and place it in the earliest place where it will fit in the developing schedule, use heuristic X to choose an operation from the b-th uncompleted job, and place it in ...", and so on.

We use the abbreviation FH(X) in referring to this strategy, in respect of some particular heuristic X. The simple heuristics we will refer to in this paper are the following eight. In each case, the set of possible operations to choose from are those in the 'current job'. This 'current job' is that represented (via the circular list of uncompleted jobs) by the allele in the chromosome which is being interpreted at this step.

LPT: choose the operation with largest processing time, breaking ties according to an *a priori* ordering over the operations.
SPT: choose the operation with shortest processing time, breaking ties according to an *a priori* ordering over the operations.
EF-LPT: Let t be the earliest time at which an operation can be scheduled, and let S be the set of operations that can be scheduled at t. Simply apply LPT to the operations in S.
EF-SPT: As above, but using SPT instead of LPT.
EF-BTR: As above, but simply choosing randomly from the set S.
SG-LPT: Let G be the set of operations that can be placed in a gap in the schedule; that is, those operations which fit inbetween two already scheduled operations on the same machine. Apply LPT to the operations in G. If G is empty, proceed as with LPT.
LRG: Choose the operation from G which leaves the longest amount of time in its gap, breaking ties randomly. If G is empty, then simply use PT.
SRG: Choose the operation from G which leaves the shortest amount of time in its gap, breaking ties randomly. If G is empty, then simply use LPT.

For example, in later experiments using FH(LPT), this refers to the fixed-heuristic hybrid method, with LPT being the heuristic used in this case.

3.4 Evolving heuristic choice

Finally, note that there is no good reason to rely on a fixed heuristic for each choice of operation while building a schedule. Indeed, it is quite easy to see that varying the choice of heuristic according to the particular job being processed, and also according to the particular stage in the schedule building process, may make more sense. It is hard to find some principled *a priori* method for making these varied choices, but we can implement a simple adaptive strategy by extending our basic chromosome representation as follows. A chromosome $abcd\cdots$ now means: "use the a-th heuristic to choose an operation from the b-th uncompleted job, and place it in the earliest place where it will fit in the developing schedule, use the c-th heuristic to choose an operation from the d-th uncompleted job, and place it in ...", and so on. We dub this method 'EHC', for 'Evolving Heuristic Choice'. Alleles of genes which are interpreted as heuristic choices (eg: odd-numbered genes in the above example) range through the number of available heuristics; in the experiments described later, the set of possible choices are the eight described earlier. We have found it beneficial for these alleles to be preferentially set to one in particular of these choices (LPT for most problems) in the initial generation, thereafter being allowed to vary via mutation and recombination. In the next section then, when we refer to the use of a *particular* heuristic in association with EHC, we simply mean that the *initial generation* of chromsomes have their heuristic choice alleles set to this heuristic, but are allowed to vary from then on.

4 EXPERIMENTS AND RESULTS

We tested each approach on six benchmark OSSPs of sizes 4×4, 5×5, 7×7, 10×10, 15×15, and 20×20. In each case, the GA used fitness-proportionate selection based on the objective function `makespan - lower bound`, lower bounds being provided in [12]. Elitist generational reproduction was used; uniform crossover was used, applied adaptively. That is, the crossover rate began at 0.8, and was reduced by 0.0005 after each generation down to a minimum of 0.3. Two children were produced from each crossover; these children, or the parents (when crossover was not applied) were each mutated with a fixed probability of 0.5; mutation involved swapping two randomly chosen genes. In each case, the results report the average of the best makespan found from each of ten trial runs of a maximum of 1000 generations, and the best makespan found overall. Convergence typically occured very quickly (sometimes in the initial generation) on the 4×4 problem; on larger problems, convergence ranged from an average of around 30 generations for the 5×5 problem to an average of 350 generations for the 20×20 problem. The population size was 200 in each case.

Table 2 shows results for the smaller three benchmarks. The 'Previous Best' row gives the best previously known solution. In the 4×4 and 5×5 cases, these are known to be global optima.

Table 2. Results on small benchmark OSSPs

	Benchmark OSSP (jobs $\times$ machines)		
	4×4	5×5	7×7
Previous Best	193	300	438
JOB+OP			
Mean	194.4	308.5	454.1
Best	193	302	441
Fixed Heuristic			
Mean (EF-SPT)	193.4	303.9	449.7
Best (EF-SPT)	193	301	445
Mean (EF-LPT)	211.0	312.2	449.8
Best (EF-LPT)	211	305	443
Mean (EF-BTR)	195.6	305.6	448.9
Best (EF-BTR)	195	301	436
Evolving Heuristic Choice			
Mean (EF-SPT)	193.0	305.0	449.7
Best (EF-SPT)	193	300	441
Mean (EF-LPT)	194.3	307.6	444.9
Best (EF-LPT)	193	305	435
Mean (EF-BTR)	194.1	305.3	449.8
Best (EF-BTR)	193	300	435

The most striking aspect of these results was that EHC yielded a better result, 435, than any previously found (that we know of) on Taillard's 7×7 benchmark ; the previous best for this was reached by tabu search [12]. Note also that FH(EF-BTR) also improved on this previous best. More generally, it appears that the methods incorporating SPT (as fixed in FH, or *initially* fixed in EHC) are best on the two smallest problems while LPT shines through on the larger problem. We find that this reliably extends to problems larger than 7×7, and hence only incorporate LPT (as the fixed or initially fixed choice) in the experiments to follow. All of the FH and EHC methods improved on the JOB+OP trials, showing a clear benefit for some form of hybridisation.

In table 3, we compare JOB+OP, FH(LPT), and EHC(LPT) for the three larger benchmarks. Note that in this case the previous best results are known to be global optima for the 15×15 and 20×20 cases [12].

Table 3. Results on large benchmark OSSPs

	Benchmark OSSP (jobs $\times$ machines)		
	10×10	15×15	20×20
Previous Best	645	937	1155
JOB+OP			
Mean	690.7	968.9	1244.5
Best	668	951	1224
Fixed Heuristic (LPT)			
Mean	662.7	942.9	1163.7
Best	646	937	1155
Evolving Heuristic Choice			
Mean	660.1	940.1	1167.7
Best	641	937	1156

The most striking result in this case is the discovery of a new best result for the 10×10 benchmark, which again was obtained using the EHC method. Note also that these results further underline the quality of a hybrid approach as compared to that of the 'pure GA' JOB+OP method. Beyond these observations we cannot really discern any clear indications as to the relative quality of FH and EHC on the two larger benchmarks.

5 DISCUSSION

The approach we describe provides excellent results on difficult benchmark problems. Although this is no guarantee that the approach will generalise successfully to real problems, and/or perform just as well on different and larger benchmarks, it is clearly a promising enough basis for continued research along these lines.

It is particularly encouraging that even the best results were achieved with an essentially simple technique, improvements on which can be readily imagined. This augments a continuing theme in GA research literature, which shows that GAs begin to compete closely with or outperform other known methods on some problems when successfully hybridised [10, 11, 8]. Further work is under way to study more sophisticated heuristics and hybridisation strategies.

In the context of the interplay between the GA and the heuristics, these results appear counter to findings like those of [1], which suggest that the more search done by the GA at the expense of the heuristic, the better in terms of final solution quality, though probably at the expense of time. Our results, and those of other authors in other applications, tend to show the opposite: better quality results arrive through hybridisation with a heuristic, with little extra time cost. Reconciliation of such counter observations are readily found however, by recognising that most generalisations we make from necessarily small forays into the space of possible experiments are at the mercy of being overturned by further such investigation. Better solution quality may well have arrived here through a 'pure' GA approach (such as JOB+OP) but only at a rather extreme cost in time; eg: for JOB+OP to compete in terms of solution quality with EHC may well be possible, but perhaps only if we use far larger population sizes and consequently wait far longer for convergence. Bagchi *et al*'s notion makes intuitive sense if we consider that it allows the GA full rein over the space of possible solutions, rather than searching a contracted space as is effectively done in most hybrid methods. However, this 'expansion' of the space that we allow the GA to survey carries with it the need for more extensive sampling and hence much larger population sizes. A hybrid GA/heuristic method thus tends to seem the better practical choice, offering a better tradeoff in terms of speed *vs* quality on most problems.

5.1 Extension to more 'real' OSSPs

The more general statement of a jobshop problem is more complex than that described here in two main ways. First, an operation has a collection of alternative process plans (it can be done on different machines), rather than the single process plan of being specified to be done on a particular machine. Let each job j_i have p_i alternative process plans. Each such plan is a distinct set of machines and associated processing times, each representing an alternative way of discharging the job. We might extend our representation to incorporate such alternatives as follows: a schedule $abcd\cdots$ means: "choose an untackled operation from the the a-th uncompleted job, using the b-th valid process plan for this job, and place it into the earliest place where it will fit in the developing schedule, ...", and so on. Here, when the schedule builder identifies the job currently referred to in the chromosome (via the heuristic choice), earlier choices in the schedule constrain the set of valid alternative process plans that are still 'live' for this job. The valid set is treated as circular, and chosen from as directed by the chromosome. There are of course several other possibilities. For example, we could use essentially the same representation as used for the simpler OSSP, but change the interpretation to: "heuristically choose a valid process plan from the a-th uncompleted job and then heuristically choose an operation from this plan, and place it into the earliest place where it will fit in the developing schedule, ...", and so on. This involves the addition of a heuristic to choose the process plan as well as choose an operation. Possibilities for the heuristic which chooses the process plan are easily imagined. For example, we might choose the plan for which the total processing time remaining is smallest.

The second important difference is that jobs have due dates which need to be considered, and also relative precedence. This can be dealt with in our approach simply by incorporating these considerations into the fitness function. That is, the fitness measure is a combination of the makespan of the schedule and the extents to which job precedences are honoured and due dates are met. Alternatively, preference and due date information may be readily incorporated into a heuristic. For example, instead of a heuristic which chooses the job with the largest processing time, we might choose the job which maximises some function of processing time and the extent to which its due date is met.

Hence, various possibilities are apparent for extending the approach to deal with more general problems. Such has been reported, for example, in the context of highly generalised manufacturing scheduling problems [7], although this did not report on the hybridisation of the GA with simple heuristics (chromosomes were much more direct representations of schedules). Our main point here is to show how our approach readily allows for extensions which will allow it to cope with problems of the fully general kind found in real machine shop environments, while still retaining its basic flavour, and hence retaining the presumed source of its success. It may not be immediately apparent that the success we demonstrate on simplified benchmark OSSPs will carry over to effective performance on more complex problems in an extended approach, but there is no apparent reason to be too skeptical of this possibility. Further work along these lines will be reported in due course.

5.2 Conclusion

We have presented an approach to the OSSP which performs very promisingly on benchmark OSSPs, twice outperforming previous reported attempts. We discussed how the approach may be extended to deal with more realistic problems; the simplicity of the approach, its apparent success, and the evident potential for much further improvement and extension, seem to render it a promising method warranting further research. Ultimately, of course, comparisons with other AI- or OR- based methods will be instructive. Also, the approach as presented fails to meet some possible needs which schedule managers may have in machine shop environments; eg: there is no clear way in which *rescheduling* can be addressed, other than by redefining the problem as necessary and running the GA from scratch; a more sophisticated technique however would be one which made use of information gained during formation of the previous schedule, which makes rescheduling a potentially very speedy process.

Finally, it should be noted that the GA configuration used in the experiments here is not optimal. Continuing GA research reveals variants and techniques that GA application researchers will find rewarding to heed. For example, [2, 3] both describe spatially-oriented selection strategies which seem to consistently outperform others, while [4] describes, among other things, reinitialisation strategies which can enhance overall robustness and reliability.

5.3 Notes

The benchmarks used here can be obtained via [9]. The OR library referred to in [9] is an electronic library of benchmarks for a wide range of OR problems. Researchers wishing to compare with our results will need to know that the problems referred to here are each the *problem No. 1* of their specified size and kind. Alternatively, problem data may be obtained directly from the authors, as can the details of the schedules found here which improve on previous best known results.

ACKNOWLEDGEMENTS

We are grateful to two anonymous referees for helpful and constructive comments on an earlier version of this paper. Thanks too to the UK Science and Engineering Research Council for support of Dave Corne via a grant with reference number GR/J44513.

REFERENCES

[1] Sugato Bagchi, Serdar Uckun, Yutaka Miyabi, and Kazuhiko Kawamura, 'Exploring problem-specific recombination operators for job-shop scheduling', in *Proceedings of the Fourth International Conference on Genetic Algorithms"*, eds., R.K. Belew and L.B. Booker, pp. 10–17. San Mateo: Morgan Kaufmann, (1991).

[2] Robert J. Collins and David R. Jefferson, 'Selection in massively parallel genetic algorithms', in *Proceedings of the Fourth International Conference on Genetic Algorithms*, eds., R.K. Belew and L.B. Booker, pp. 249–256. San Mateo: Morgan Kaufmann, (1991).

[3] Yural Davidor, 'A naturally occuring niche & species phenomenon: The model and first results', in *Proceedings of the Fourth International Conference on Genetic Algorithms*, eds., R.K. Belew and L.B. Booker, pp. 257–263. San Mateo: Morgan Kaufmann, (1991).

[4] Larry J. Eshelman, 'The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination', in *Foundations of Genetic Algorithms*, ed., G. Rawlins, 265–283, Morgan Kaufmann, (1991).

[5] Hsiao-Lan Fang, Peter Ross, and Dave Corne, 'A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems', in *Proceedings of the Fifth International Conference on Genetic Algorithms*, ed., S. Forrest, 375–382, San Mateo: Morgan Kaufmann, (1993).

[6] Teofilo Gonzalez and Sartaj Sahni, 'Open shop scheduling to minimize finish time', *Journal of the Association for Computing Machinery*, **23**(4), 665–679, (October 1976).

[7] P. Husbands and F. Mill, 'Simulated co-evolution as the mechanism for emergent planning and scheduling', in *Proceedings of the Fourth International Conference on Genetic Algorithms*, 264–270, San Mateo: Morgan Kaufmann, (1991).

[8] Jr. James D. Kelly and Lawrence Davis, 'Hybridizing the genetic algorithm and the k nearest neighbours classification algorithm', in *Proceedings of the Fourth International Conference on Genetic Algorithms*, eds., R.K. Belew and L.B. Booker, pp. 377–383. San Mateo: Morgan Kaufmann, (1991).

[9] Beasley J.E., 'OR-library: Distributing test problems by electronic mail', *Journal of the Operational Research Society*, **41**, 1069–1072, (1990).

[10] Si-Eng Ling, 'Intergating genetic algorithms with a prolog assignment problem as a hybrid solution for a polytechnic timetable problem', in *Parallel Problem Solving from Nature, 2*, eds., R. Manner and B. Manderick, 321–329, Elsevier Science Publisher B.V., (1992).

[11] Colin Reeves, 'Hybrid genetic algorithms for bin-packing and related problems', Technical report, School of Mathematical and Information Sciences, Coventry University, (1994). submitted to *Annals of OR*.

[12] E. Taillard, 'Benchmarks for basic scheduling problems', *European Journal of operations research*, **64**, 278–285, (1993).