Genetic Programming and Evolvable Machines, 6, 191–220, 2005
© 2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

Evolutionary Scheduling: A Review

EMMA HART E.Hart@napier.ac.uk
PETER ROSS P.Ross@napier.ac.uk
Napier University, Scotland, UK

DAVID CORNE D.W.Corne@ex.ac.uk
University of Exeter, UK

Submitted August 18, 2003; Revised May 12, 2004; Accepted May 26, 2004

Abstract. Early and seminal work which applied evolutionary computing methods to scheduling problems from 1985 onwards laid a strong and exciting foundation for the work which has been reported over the past decade or so. A survey of the current state-of-the-art was produced in 1999 for the European Network of Excellence on Evolutionary Computing EVONET—this paper provides a more up-to-date overview of the area, reporting on current trends, achievements, and suggesting the way forward.

Keywords: scheduling, evolutionary algorithms

1. Introduction

Scheduling can loosely be described as allocating resources in order to complete a number of tasks economically within given constraints. There are many kinds of scheduling problems: arranging grocery deliveries or rubbish collections; creating staff work rotas for hospitals or for bus or train services; arranging examination timetables in universities; arranging for a set of items being constructed in a factory to each visit certain work cells; scheduling equipment maintenance; and so on. The variety is enormous; the common threads are a set of tasks to achieve, a set of resources needed, a set of constraints that must be obeyed ('hard constraints') and perhaps another set of constraints that it would be good to have obeyed if possible ('soft constraints'), and often a set of costs associated with actions. For example, in scheduling equipment maintenance there is a cost associated with any unavailability of the equipment; in scheduling tasks through a work cell there may be costs associated with setting up that work cell for the task, which may depend on the previous task, such as changing paint type in a paint cell, or changing cutters in a machining cell. As a basic example of scheduling, consider *job-shop scheduling problems*—these have been much studied in the academic literature, usually in an idealized form that omits nearly all the messy real-world constraints. Figure 1 shows a simple example, in which there are 6 jobs each consisting of six tasks—a task consists of that job visiting a certain machine for a certain length of time. So, for example, job 2 must first visit machine 2 for 8 units of time, and next machine 3 for 5 units of time, and so on. In this problem, it so happens that each job must visit all six machines, in some order that is job-dependent; but this is not a necessary feature of job-shop

	Tasks					
Job	(m,t)	(m,t)	(m,t)	(m,t)	(m,t)	(m,t)
1:	3,1	1,3	2,6	4,7	6,3	5,6
2:	2,8	3,5	5,10	6,10	1,10	4,4
3:	3,5	4,4	6,8	1,9	2,1	5,7
4:	2,5	1,5	3,5	4,3	5,8	6,9
5:	3,9	2,3	5,5	6,4	1,3	4,1
6:	2,3	4,3	6,9	1,10	5,4	3,1

Figure 1. A simple job-shop scheduling problem.

problems in general. Figure 2 shows a solution in which the *makespan*—the length of time from start of the first task to the end of the whole problem—is as short as possible.

It has long been known that even very simple versions of scheduling problems are NP-hard. For example, in job-shop scheduling, if the number of machines is two, and the number of tasks per job are both restricted to two, then the problem is computationally easy, but as soon as we allow either three tasks per job or have more than two machines, then the problem is NP-hard. Similarly, whenever the number of jobs is more than two, the problem is NP-hard. Of course real-world examples of job-shop scheduling problems are vastly more sizeable than this. Vaessens et al. [116] provide a good roundup of these and related results.

These and other academic results and research in scheduling generally concern optimisation of makespan. However, the makespan is rarely of commercial interest, because it is unusual to have a scheduling problem all of whose details are known at the start, which doesn't change unexpectedly, and which has a clear end. Such problems do still occur, for example in documentation approval processes, but it is more usual to have

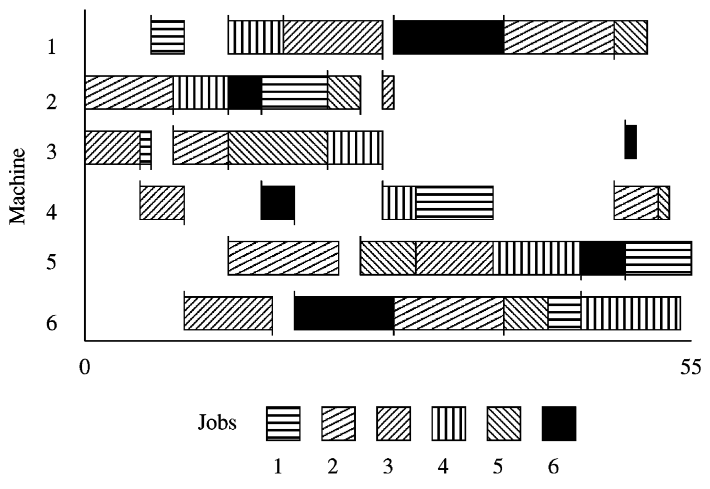

Figure 2. An optimal answer.

dynamically-changing problems, in which the measure of quality is something like *weighted tardiness*, an importance-adjusted sum of how late the jobs are beyond their desired completion date. Even so, attempting to solve the above problem by hand should illustrate why computer assistance is so valuable in this domain, and why so much research has been directed towards solving such problems. There are clearly many and varied applications that can be termed as scheduling—the aim of this paper is to describe the trends in the application of evolutionary computing techniques to the broad area of scheduling, and provide a summary of the current state-of-the-art in the field. The paper deliberately does not consider the application of evolutionary techniques to the area of timetabling—although clearly timetabling is a subclass of scheduling problems, it is a large field in its own right and it is the authors' intention to consider it in a separate paper. The review is deliberately focussed on the field of Evolutionary Computation—other authors, for example Jain and Meeran [61] have provided in-depth comparative surveys on solving particular scheduling problems, e.g. job-shop scheduling, which compare and contrast a variety of techniques; this is not the concern of this paper which attempts to evaluate how successful the field of Evolutionary Computation has been at tackling such problems. Early pioneers in the field of evolutionary computing were quick to realise the potential of the technique as a tool for tackling some ofthe standard scheduling problems that had long been the subject of research in the Operations Research community. Seminal works such as [5, 18, 26, 58, 88, 109, 112, 121, 122] introduced key ideas such as a range of chromosome representation strategies and specific genetic operators. These papers provided early evidence for the usefulness of of evolutionary techniques in the scheduling domain, finding them better than rival techniques in certain circumstances. This paper presents a review of the field as it is today. It attempts to assess how far research has advanced since the early scheduling papers, define the current state-of-the-art, and finally make suggestions for the way forward.

1.1. Guide to paper

Sections 2 to 4 broadly review the types of scheduling problem tackled by evolutionary algorithms. There is no natural and obvious taxonomy for categorising scheduling problems in general, hence we loosely divide the field into three areas; academic problems, reality-enhanced academic problems, and finally 'idiosyncratic' problems. The latter area includes examples of real-world applications in which the trend seems to have been to take ideas from much of the academic work in the past few years, and adapt these notions towards use in very problem specific ways. Whilst some essential ideas are often retained from earlier work, the problem specific details in particular cases seem to call for the use of specialised methods in each case, not easily transferable or generalisable to other scheduling problems. Section 5 gives a technical overview of the representations and operators used in current state-of-the-art implementations of EAs applied to scheduling problems. Section 6 discusses some emerging techniques and refinements to the standard EAs. Section 7 presents an overview of other biologically and evolutionary inspired approaches to scheduling, and finally, the paper is concluded in Sections 8 and 9 which identify the current weaknesses in the field today and makes suggestions for the future direction of evolutionary scheduling research.

2. Academic problems

The application of evolutionary computing methods to the established academic benchmark scheduling problems is still prevalent in the literature. Benchmark problems usually consist of a number of jobs, each containing a fixed number of operations or tasks, which have to be performed on one of more machines. Varying the relationship between the number of machines, jobs and operations, and the rules governing how operations are scheduled produces a number of different categories of problem; single machine shop (SMP), Parallel Machine Shop (PMP), Flow-Shop (FSP), Job-Shop (JSP) and Open-Shop (OSP). In the SMP, each job has a single operation to be performed on the single machine existing in the shop, thus the scheduling task is to find the optimal sequence in which the jobs must be processed. In the PMP, each job has a single operation but there are m machines working in parallel. A job can be processed on several (or possibly any) of the machines. The FSP contains m machines, and each job consists of a strictly ordered sequence of operations. The machine order for all jobs is identical. Finally in JSP there are m machines and n jobs, and each job has its own processing order through the machines, which may bear no resemblance to that of any other job. An extension of this class is the open-shop, OSP, in which there is no such restriction of order within in each job. A general description $n/m/A/Obj$ can be applied to the above academic problems, in which n indicates the number of jobs, m the number of machines, A indicates the flow pattern of jobs, and Obj the objective by which the quality of the schedule is measured. A can have value P to indicate permutation flow, that is the order of processing of jobs is identical on each machine (as in flowshop problems), and a value of G indicates arbitrary flow (as in jobshop problems). There are many possible ways in which the quality, Obj, of a schedule can be meaured—for example, $C_{\max}$ indicates makespan, C_{sum} indicates the problem of minimizing the sum of the completion times of each job. Tables 1 and 2 provide lists of the most frequent measures of which quality is measured in academic problems, adapted from French [44], all of which must be *minimised* in order to provide an optimum solution. Other criteria exist—for example, the criteria in the two tables just mentioned can all be weighted by individual jobs. Other examples based on inventory and utilisation costs include the mean number of jobs waiting for machines, the mean number of unfinished jobs, or the mean number of complete jobs etc. However, the tables represent the most commonly applied criteria.

Table 1. Schedule quality measures based on complete time.

Quality measure	Symbol	Interpretation
Maximimum complete time	$C_{\max}$	(Makespan): The cost of a schedule depends on how long the processing system is devoted to the entire set of jobs.
Mean complete time	$\bar{C}$	The schedule's cost is directly related to the average time it takes to finish a single job.
Maximum flow time	$F_{\max}$	The cost is directly related to the longest job.
Mean flow time	$\bar{F}$	The cost is directly related to the average time it takes to process a single job.

Table 2. Schedule quality measures based on due-date.

Quality measure	Symbol	Interpretation
Maximum lateness	$L_{\max}$	The schedule'c costs is directly related to its latest job.
Mean lateness	$\bar{L}$	The cost is directly related to the average difference between complete times and due-dates for all jobs. Early jobs in effect contribute reward, due to negative differences.
Maximum tardiness	$T_{\max}$	The cost is directly related to the latest job that completes after its due-date.
Mean tardiness	$\bar{T}$	A schedule's cost is directly related to the average lateness for all jobs, where early jobs are considered to have a late time of 0.
Number of tardy jobs	NT	The schedule's cost depends on the number of jobs that complete after their due date.
Maximum earliness	$E_{\max}$	A schedule's cost is directly relatedto the earliest job that completes before its due-date.

2.1. *Job-shop scheduling*

Job-shop scheduling perhaps receives most attention from the evolutionary community, by a plethora of different approaches. There are two versions of this class of problem. In the *static* job-shop, all jobs are available for processing at the start of the operation. In the *dynamic* version of the problem—a situation which bears much more resemblance to real-world scheduling problems—jobs arrive throughout the scheduling process. This can be either deterministic—the arrival date of each job is known at the start, or *stochastic*—arrival dates are stochastically generated. Cheng et al. [16] provide an in-depth survey of representations used in genetic algorithms to solve JSSPs which fall into two classes; those that use *direct* representations and those that use *indirect* representations. In a direct representation, the schedule itself is directly encoded onto the chromosome. On the other hand, many representations, described as indirect, concentrate on specifying the order in which elements of the schedule should be considered for placement into the schedule. Table 3 sub-classifies each of these classes into a number of different types of representation, with references which provide examples of the technique.

A more in-depth survey of representations employed more generally in scheduling is given later in this paper, however the reader is referred to Cheng et al. [16] for a detailed

Table 3. Representations used in JSSPs.

Direct representation	Indirect representation
Operation based	Preference-list based
Job based	Priority-rule based
Job-pair relation based	Disjunctive-graph based
Completion-time based	Machine based
Random-keys	

discussion of each of these representations with respect to job-shop scheduling. Vásquez et al. give a comparison of the performance of the current state-of-the-art in evolutionary algorithms for static JSSPs in [118]. Their paper identifies four very different algorithms and compares their performance across a set of benchmark problems. The algorithms in question are *OBGT* due to [117], *HGA*, due to [50], *THX*, from [76] and finally GA3 [79]. OBGT and THX utilise direct, operation-based representations. In OBGT, the chromosome represents a permutation of the jobs for each machine, and each operation also contains its start time. Order-based operators are applied, and the Giffler and Thompson [46] and Non-Delay algorithms used to generate schedules. Local search is applied to all offspring using the *Grabowski critical neighbourhood* with a steepest descent technique to improve generated schedules. The Giffler & Thompson algorithm is a systematic approach to generating active schedules, i.e. schedules in which no operation can be completed earlier without delaying other operations. It is very common in evolutionary algorithms to find the G&T algorithm [46] utilised in some form, therefore for reference it is given in Figure 3. The Grabowski neighbourhood is one of a number of local search neighbourhoods observed in GA hybridisations (see Section 6.1 later in this paper) in which a move is defined by repositioning an operation to either the front or the rear of a critical block. In THX, each individual encodes for each operation (in index order) its start time in a feasible schedule, and uses genetic operators which operate at the schedule level to perform crossover and mutation operations. Thus, a crossover denoted as time-horizon-crossover exchanges operation-sequences from parent individuals such that temporal relations among operations are preserved. The mutation operator selects two operations on the critical path and reverses these operations. The operators are again inspired by the Giffler & Thompson (G&T) algorithm. Finally, GA3 uses an order-based representation consisting of a permutation of operations, considering this to be a natural representation of the problem. A permutation of job identifiers rather than actual tasks is used to avoid producing infeasible schedule due to precedence constraints between tasks. For example, a chromosome 322231113

```
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 manchine on which the minimum 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.
```

Figure 3. Giffler and Thomsom algorithm.

denotes that the 1st operation of job 3 is scheduled, followed by the 1st,then 2nd, then 3rd operations of job 2 etc. Their GA uses a structured population model based on a toroidal grid, with local mating. Reproduction is controlled using *attitude inheritance*—essentially this adaptively determines whether an individual undergoes crossover with another individual in its neighbourhood, mutation or otherwise performs no active operation, therefore avoiding possible replacement by offspring, depending on a individuals immediate environment. HGA, (heuristically-guided-GA) uses an indirect representation which is a modified version of a priority-rule based representation. In HGA, each gene in a chromosome of length $(j * m)$ represents a pair of values, (*method, heuristic*), where $method$ denotes a choice of two algorithms that should be used at each iteration of a scheduling algorithm to calculate a set of schedulable operations, and *heuristic* denotes the priority dispatch rule to be used to select an operation from that set. A priority dispatch rule is simply a heuristic for selecting an operation to schedule when a machine becomes free. Panwalker and Isander [98] listed more than one hundred such rules, and later [11] published a state-of-the-art survey of dispatching rules in manufactuting job-operations. Examples of commonly used rules are given in Table 4. The two scheduling algorithms used are the G&T algorithm which produces active schedules, and the Non-Delay algorithm. The Non-Delay algorithm is a modified version of the G&T algorithm in which a machine is never kept idle if some operation is able to be processed, and results in schedules which are a subset of the active schedules generated by the G&T algorithm. Vásquez and Whitley [118] compare the performance of the above algorithms on the classical Fisher and Thompson benchmarks (FT06,FT10,FT20), Yamada and Nakano (YN1,YN2,YN3,YN4), Lawrence (LA21,LA27,LA29,LA38) and Taillard (TA051–Ta060), all available from the OR-library [7]. A further set of job-correlated problems (i.e. problems in which jobs have correlated processing times across machines) and machine-correlated problems (in which job processing

Table 4. Priority Dispatch Rules: Assume t is the current time, r_i is the arrival time, d_i is the due-date, T_i is the total processing time, and n_i is the number of remaining operations of the ith job. Futhermore, for each operation j of each job i, then p_{ij} is the processing time, od_{ij} is the operational due-date, mod_{ij} is the modified operational due-date, s_{ij} and is the slack time.

Rule	Description
RND	Randomly, select amongst operation with equal possibility
FASFS	First-arrive-shop-first-serve , select operation with smallest r_i
SPT	Select operation with shortest processing time, p_{ij}
WSPT	Seclect operation with weighted shortest processing time, w_i/p_{ij}
LPT	Select operation with largest processing time p_{ij}
LWKR	Least work remaining, R_i
WLWKR	Weighted least work remaining, w_i/R_i
LTWK	Least total work, select operation with smallest T_i
WTWORK	Select operation with largest w_i/T_i
EGD	Earliest global due-date, select operation with smallest E_i
EOD	Earliest Operational Due-Date, choose smallest $od_{ij} = r_i + (d_i - r_i) * R_i/T_i$
EMOD	Earliest modified operational due-date, $mod_{ij} = larger(od_{ij}, t + p_{ij})$
MST	Minimum slack time, smallest $s_{ij} = d_i - R_i - t$
S/OP	Slack-per-operation: smallest s_{ij}/n_i
CR	Critical ratio, largest $R_i/(d_i - t)$

times are similar on a particular machine) were also tested. The authors conclude in general, OBGT is a viable alternative for solving JSSP although OBGT results are somewhat disperse in that there is a high variance among makespan values, and that its scalability to large problems has not been determined. Regarding HGA [118], note "HGA is a very good near-optimal strategy and it does not possess scalability problems. HGA consumes less computational resources than OBGT and the variance among HGA results is smaller than the variance obtained by the rest of the GAs (only GA3 produces less variance in some cases and HGA is competitive in terms of variance to NS). HGA is a good candidate to be applied in real-time scheduling and in dynamic scheduling applications." GA3 was found to be an extremely competitive GA for solving JSSP; it was only beaten in the Yamada and Nakano benchmark. However, it was not tested on the job-correlated and machine-correlated problems. In these problems, OBGT produced the best results and in the case of job-correlated problems THX turned into a good alternative for small problems, although it did not perform well in OR-library benchmark problems. In a separate paper, Vásquez and Whitley [117] also compared the performance of THX-P, OBGT and HGA on a series of benchmark dynamic JSSPs, along with another GA implementation due to Fang [40] and to Priority Dispatch Rules. These results show that although the best results across all problems were obtained by THX-P, OBGT and HGA, a pairwise t-test failed to show any difference between any of these algorithms. Further analysis using a chi-squared test showed that the number of times that OBGT was able to find the best known solution was significantly better than HGA or TGHX-P, and that there was no difference between the latter two algorithms. This comparison nicely illustrates the diversity employed in evolutionary scheduling algorithms. All four state-of-the-art algorithms utilise very different representations and operators, and none emerges as the absolute winner across all problems. Although this is perhaps unsurprising in light of the no-free-lunch theorem [123] it does illustrate the need for some more thorough research to investigate why certain algorithms work better on certain problems, i.e. to attempt to produce a mapping of algorithm performance to problem features. One point of commonality across three of the algorithms—THX, HGA and GA3—is that they are all implemented in parallel, therefore this raises the question as to whether this is one implicit feature of the evolutionary algorithm that could be exploited more often in evolutionary algorithms in order for them to gain a competitive advantage over other types of search algorithm.

2.2. *Flowshop problems*

Rather fewer applications of evolutionary techniques relate to flowshop scheduling problems. Reeves et al. [102] performed an empirical analysis of two of the Taillard benchmark problems in order to produce a GA that efficiently exploited the structure of the landscapes found. The flowshop problems were found to induce a 'big-valley' structure, with local optima occurring relatively close together and to a global optimum. This resulted in the technique described as path relinking, which is based on the hypothesis that if two locally optimal solutions exist then tracing a path from one solution to another may find another local optimum. The resulting GA finds impressive solutions, in fact beating the best known results of Nowicki and Smutnicki [93] (obtained using tabu-search) on the 50×20

problem. This approach still represents the state-of-the-art as far as EA approaches to FSPs are concerned. Esquivel et al. [37] compare a number of EA approaches to FSPs however, unfortunately a different set of instances of the Taillard problems is used to those used by Reeves therefore a direct comparison cannot be performed between the published algorithms. However, Esquivel et al. compare a 'standard' EA using a single point crossover operator (SCPC) to a *multi-parent* approach (see [36] for more details) in which offspring creation is based on a much larger sample of the search space which increases diversity and helps avoid premature convergence. Esquivel et al.'s proposal used the PMX crossover to build offspring for sets of parents and the algorithm is hybridised with a heuristic known as the Nawaz-Enscore-Ham heuristic (NEH) to give an algorithm known as MCPC-NEH. This heuristic is widely regarded as the best-performing heuristic for the FSP [113] and works as follows [119]:

1. Order the n jobs by decreasing sums of total job processing times on each of the machines.
2. Take the first two jobs and schedule them so at to minimise the partial makespan as it there were only 2 jobs.
3. for $k = 3$ to n do
   - insert the k-th job into the location in the partial schedule so as to minize the partial makespan.

The heuristic essentially gives higher priority to jobs with higher processing time on all machines. When the MCPC-NEH and SCPC algorithms were compared to an ant algorithm also hybridised with the NEH heuristic, A-NEH, on a set of the smaller Taillard instances, then MCPC-NEH was observed to perform the best. However, as the authors themselves note, MCPC-NEH requires a large programming effort and needs higher computational effort, so further research is required in this area, especially if the algorithm were to be used for real-world applications.

2.3. Open shop problems

OSP have attracted the least attention of the academic Job-Shop scheduling problems. Fang [41] proposes a method named 'Evolving Heuristic Choice'. Similar to HGA discussed above, the GA uses a representation containing three components, namely 'ABCD..' and 'abcd..' meaning 'apply heuristic rule A to decide which operation of the ath uncompleted job to use and put it in the earliest place possible in the developing schedule. The method was capable of beating many of the previous best solutions on a set of benchmark problems obtained from [7] and provided new upper bounds on many of the problems.

3. Reality enhanced academic problems

The growing trend towards what we term 'reality-enhanced' academic problems certainly has seeds in quite early work. For example, Bruns [13], Husbands and Mill [58] and Husbands et al. [59] all looked at problems with an extra layer of complexity beyond standard 'job-shop' by virtue of having alternative process plans for each job. That is, there may be different ways a job can be processed through a factory, using a different 'route' of machines or workstations. This is certainly the case in many real work problems, most clearly, for example, when there are collections of identical machines which can perform the same types of task. Also, Cartwright and Tuson [15], Fang [38] and Fang et al. [41] all looked at the issue of 'rescheduling', which is more commonly known in the scheduling community as 'dynamic scheduling'. This reflects the real-word factory environment situation in which jobs arrive ready for processing intermittently and continually during the day. Critically, there is in many cases no real value in finding an optimal or near-optimal schedule regarding the jobs currently available for processing, since jobs arriving during that schedule's operation will undermine the process. The last main 'reality enhanced' issue concerns 'stochastic' flow shop problems. This is where we recognise that the machine processing times and also the job-arrival times which generally occur in benchmark or test data are potentially quite inaccurate. For example, we might estimate 30 minutes for varnishing a particular manufactured wooden item, but find in practice that it varies between about 15 and 60 minutes. Recent evolutionary algorithm based work which takes this into account includes [2, 75, 76]. A further type of 'reality-enhanced academic problem' route has been a trend for researchers to create new definitions of job-shop style problems with additional types of constraints and features which reflect a particular real-world application area. Alternatively, there is a trend to use evolutionary methods on well known 'abstract' problems which are importantly different from simple job shop scheduling. Such problems come along with their own benchmarks, or problem generators, and involve many constraints and features which differentiate them from pure job-shop, although they remain in the 'academic benchmark' class. For example, Ramat et al. [100] describe an evolutionary approach to the Multiple Resource Constrained Scheduling Problem (MRCSP); this is a well known problem in the scheduling research community, but seemed not to have been tackled with GAs prior to 1997. In the MRSCP, jobs need *resources* rather than machines. In fact, the 'machine' in a classic job-shop problem is just a particularly simple type of resource: one which is available all the time, and which can process precisely one task at a time. The MRCSP generalises job-shop problems with resources which have time windows of availability and capacity. For example, a high performance computer may be available at certain times and able to process more than one task at a time, depending on the particular loads each task would place on it. Ramat et al. find that a GA approach, tested on randomly generated examples of the MRSCP, fares significantly better than a previously published simulated annealing approach to the same problems, and also better than simple priority-rule based methods. Another example is given by Norman & Bean's recent account of a scheduling problem arising in environments with 'Multiple Spindle CNC Machines' [92]. This nicely reflects the fact that new types of scheduling problem will naturally arise as a result of advances in factory machinery. In this case the development in question is the emergence of 'Parallel Machine Tools' (PMTs); these contain multiple spindles which can hold multiple workpieces concurrently in multiple locations on the machine. The technology thus opens up more potential 'routes' for parts through the different operations and locations on such a machine. Norman & Bean formulated this new scheduling problem in [92], generated random test data (but based on realistic ideas of the numbers and types of parts that would typically need to be processed on such machines) and compared two types of GA. Their best results were from a method which hybridised a simple GA approach with an appropriate heuristic rule. The GA evolved an initial sequence of operations against the main precedence constraints, while a heuristic rule was used to modify subsequences within this main sequence according to constraints concerned with tool assignment.

4. Idiosyncratic problems

The final direction we extract in our broad view of current trends and progress is that of 'Idiosyncratic Real World Problems'. The title of this section is a reflection of the difficulty in finding an easy and natural taxonomy for the range of problems being addressed but nicely illustrates the apparent diversity of the evolutionary approach. For example, Ryu et al. [104] concern a real-world scheduling problem in shipbuilding; the assembly of the many different 'blocks' (parts of ship) which are then later combined together in hierarchically larger blocks until a complete ship results. Each task is the assembly of a particular block, and the 'machines' are teams of assemblers who work on one block at a time. Rather than an emphasis on minimising makespan, there is a hard and easily attainable maximum and minimum makespan requirement in this problem, with the main indicator of schedule quality being the way in which workloads are balanced fairly between teams. What Ryu et al. seem to use from earlier EC/scheduling work is the key notion of hybridising a simple chromosome style (in this case a k-ary string which simply allocates an assembly team (allele) to each block (gene)), with a sophisticated problem-specific schedule builder. On the other hand, other earlier work emphasising 'makespan' issues, perhaps making use of Giffler & Thompsons's algorithm, using previously developed permutation based sequencing operators [109], and so on, are not obviously relevant and consequently not used in [104]. The 'distance' between the classical job shop and this real world case is reflected in other work, such as [73], which concerned the scheduling of maintenance tasks across part of the UK National Grid. A major complicating factor here of course is that maintenance of a generator puts that generator out of action for a time, with complex consequences for supply, depending on the location of that generator in the grid. Here, the key 'real world' aspects are again nicely reflected by a discussion of the schedule quality criterion. In [73], a good schedule is one which optimises well in terms of maintaining the capacity to meet demand (which varies with location and time), minimising cost, rewarding regular maintenance (over and above necessary and statutory maintenance), and avoiding situations where the maintenance plan would require an underlying change to the existing schedule for electricity supply. Langdon solves this problem, following a pattern common in this large group of 'idiosyncratic real-world' cases, by again hybridising a relatively simple chromosome representation with a complex, problem-specific schedule builder. In summary, the scope of problems addressed in evolutionary scheduling has broadly moved from classic 'makespan-oriented' work on academic job shop and flow shop problems towards two separate recent directions. First, a return to classic job-shop benchmarks but this time considering reality-enhanced details such as dynamicity and stochasticity, and second, a blossoming of reported work on a diverse collection of real world problems. Some of the early work and progress on classic job shop problems translates into useful material for applying to the idiosyncratic collection of real world problems, but much of it doesn't, largely owing to the non-makespan nature of real world examples, and the large collection of extra constraints and objectives. What does appear to 'survive' is more in the line of an abstract overall approach.

5. Technical issues

This section surveys some of the technical aspects of evolutionary scheduling, in particular attempting to review the range of representations and crossover operators used within published applications.

5.1. Representation

As can be expected following the discussion in the preceding sections describing the range of scheduling problems tackled by evolutionary algorithms, the methods by which schedules can be represented as individuals within a population are naturally extremely diverse. Although many specific approaches have been touched upon in the preceding sections, we mention a number of generic approaches here which represent the basis of most scheduling applications. A direct approach in which the schedule itself is encoded on the chromosome [13, 76] represents some difficulties in that it requires the design of specific operators, however it has advantages in that there can be no false competition between individuals within the population (i.e. multiple, distinct individuals representing the *same* schedule), and that changes can easily encoded back to the chromosome following mutations/local search etc. It is common to encode a schedule as a permutation, and then decode the permutation using for example Giffler and Thompsons algorithm or the Non-Delay algorithm as already mentioned. Another variant of a these decoding algorithms is the *hybrid schedule*, used by Bierwith and Mattfeld [10], in which schedules are produced in a flexible way by introducing a parameter δ which can be thought of as defining a bound on the length of time that a machine is allowed to remain idle before an operation must start. The choice of the value of d depends on the problem instance, and influences the performance of the GA. An alternative to using permutations directly is to use what is called a 'random keys' representation [91, 92], namely an array of real numbers. This is converted back to a permutation by looking at sizes: the gene with the biggest value is the first, the gene with second-biggest value is the second, and so on. This has the disadvantage that many chromosomes represent the same permutation, for example any chromosome $[x, y, z]$ in which $x > y > z$ represents the permutation [1, 2, 89]. It has the apparent advantage that many familiar, easily-understood crossover and mutation operators can be used with it, such as two-point crossover and random mutation. Furthermore, an evolution strategy approach, specific for dealing with real-number arrays, could also be applied. However, it should be noted that it is still the relative size of gene values which is important, rather than absolute sizes in determining the ordering of tasks. The random-keys approach still requires decoding of the resultant permutation—[92] suggest using a parameter called the *delay factor* associated with each operation, which determines the order in which operations are scheduled. This algorithm is referred to as the move-search scheduling algorithm. Matsui et al. [78] utilise the random-keys representation in combination with the hybrid-schedule method in a version of the Parameter-free Genetic Algorithm, PfGA [106]. They propose a chromosome of length $2 \times n \times m$, where the first $n \times m$ represent random-keys, and the remaining bits represent $n \times m$ values for δ to be used with the hybrid scheduling algorithm. Unlike the work of Bierwith and Mattfeld [10] this results in an individual value for δ being evolved for every operation. Their approach is tested on 22 problems from the ORLib [7] and directly compared to Norman and Bean's results obtained using the move-sesarch algorithm. They find that their PfGA obtains better results on small problems but is worse on some of the larger problems. They also implement a parallel version of the algorithm which improves results slightly. The indirect approach of for example encoding priority-rules [33, 50] or evolving the schedule-builder itself [54] has the advantage of being amenable to easily encoding problem-specific information within the genotype, however its disadvantages are the reverse of those just listed for direct representations; mutations or alterations to the schedule are difficult to encode back to the genotype, and there may be false competition. Corne and Ogden [22] compare direct and indirect approaches on an identical problem to schedule preacher sermons at distributed churches; they compare the performance of a genetic algorithm, hill-climber and simulated annealing with both representations and conclude that for this problem, the indirect approach is superior for all three search algorithms. Interestingly, the worst-performance using the indirect representation (using the GA as the search algorithm) is still better than the best performance using the direct representation (using simulated annealing). Corne and Ogden [22] go on to suggest that although from a development point of view, the most simple approach is to employ a direct representation, the extra effort involved in producing a schedule builder for an indirect representation is worthwhile.

5.2. Crossover operators

A large range of crossover operators have been proposed, mainly due to the need to design specialist crossovers for use with permutation representations—details of crossovers specifically designed for ordering applications can be found in [99]. The most common ones observed in EA scheduling applications are outlined here. Uniform-order crossover, UOX, has the merit of preserving the position of some genes and the relative ordering of the rest. This *may* be helpful in trying to bring good building blocks together, but it depends on the nature of the problem being solved. For an example of its use, see [126]. Many alternatives exist; JOX [94] preserves the order of each job on all machines whilst creating children, taking account of dependencies amongst machines. This can result in infeasible schedules, and this a repair method must be applied to transform any infeasible offspring. MSXF, multi-step fusion crossover, Yamada and Nakano [125] contains built-in local search functionality. This focuses attention on the region between two parents in the search space, and picks up good solutions from that region. Reeves and Yamada [102] extend this idea and introduce *path-relinking* which is effectively a local search technique which exploits the structure empirically observed to exist in FSP. Path-relinking is a stochastic search for local optima along a path between two parents, and has been demonstrated to be highly effective—for example, several new upper bounds have been found for the Taillard benchmarks using this technique [102]. EDX, extrapolation directed crossover, was proposed in [105] to be used alongside JOX in solving JSPs. JOX tends to generate offspring that either exist close to parents that produced them or in an intermediate area of them, inducing convergence of the population. EDX was designed to enlarge the population using a local search procedure and hence compensate for the drawbacks of JOX. Comparison of the performance of the technique on a set of large benchmark problems is performed against the state-of-the-art GAs, and loses out in only 2 instances. When compared to other state-of-the-art approximation algorithms, it performs well, and is only clearly beaten by Yamada's combined simulated annealing/shifting bottleneck approach described in [125]. Yet another variation on the theme is subsequence exchange crossover, or SXX [70], which exchanges subsequences in parents on each machines when they consist of the same set of jobs. Again in this crossover, the G&T is applied to transform the resulting schedules into active ones. When tested on the Fisher+Thompson 10×10 benchmark, it obtained optimal solutions. In summary, a great deal of attention has been paid to improving crossover operators for the standard benchmark problems by paying attention to the form of the search space particular to these problems. It would be also fruitful to adopt this approach when designing operators for real-world problems.

6. Refining the EA

A number of refinements to the genetic algorithm have been introduced in order to enhance its performance. Some of these refinements in the context of scheduling are now reviewed.

6.1. Hybridisation with local search

An increasingly more common approach is to add local search to the EA to improve its performance, and indeed, many of the GAs previously mentioned incorporate some kind of local search strategy, e.g. [37, 79]. Yamada and Nakano [125] obtain promising results using a scheme in which the local search is part of the recombination process. One parent is used as the starting place for a 'simulated annealing' search; the other parent is used as a source of bias, to steer the search somewhat towards that parent. In a state-of-the-art survery of jobshop scheduling problems [61], Jain made the general observation that a genetic local search framework (i.e. one in which a child conceived from GA operators is transformed by a local search to the nearest locally optimal solution) always provides better results that a GA, recents publications do not necessarily back this up (for example, see the comparative work of Vásquez and Whitley [118] mentioned in Section 2.1).

6.2. Multiobjective evolutionary algorithms

Scheduling is a natural domain to which to apply multi-objective techniques, in reality a schedule can seldom be judged by a single criteria. An overview of multi-objective algorithms for scheduling problems can be found in [4]—here we give some flavour of work that has been undertaken. An early example of a practical application of multi-objective GA was by Shaw and Fleming [107] who applied the technique to production of chilled ready meals in a factory. Other real-world applications include nurse scheduling [62], ship-building [115] and scheduling and route selection for military land moves [83]. A significant amount of work has also been directed at academic problems, for example flow-shop scheduling [60] and job-shop scheduling [14]. A recent example of such work is [12], in which the permutation flowshop problem is considered with three objectives: makespan, mean flow-time, and mean tardiness. The focus of the work was to establish which was the best of a number of crossover operators for a set of instances of this three-objective permutation flowshop problem, and they found that Syswerda's order based crossover (OBX) [112] was clearly dominant, over Precedence Preservative Crossover [9] and so-called 'One-Segment Crossover' [12]. However, in general, work on multiobjective scheduling has been fairly scarce. This may partly be because we are quite unclear about the extent to which the different objectives occuring in scheduling problems are in conflict (i.e., if generally not, then specialised multi-objective approaches would be less necessary). Recent excellent work towards understanding this issue is [49], which includes a thorough correlation analysis between each of fourteen different scheduling-related objectives for flowshop problems. The correlations are calculated on the basis of randomly generated schedules, and show that objectives based on lateness and tardiness tend to be uncorrelated with other objectives, while there is a strong correlation, for example, between makespan and maximum flow time. Such work is aimed at understanding in what situations multiobjective approaches are really required. For example, if makespan and maximum lateness are both important objectives in a particular case, then this work suggests that teh problem is best treated as a proper multiobjective problem, so that solutions exploring the tradeoff between these objectives can be appropriately explored. However, if the two objectives of interest are highly correlated, there are likely to be few such tradeoff solutions, and a single-objective optimisation approach should be adequate.

6.3. Coevolution

An area from EA research which has been little used in the scheduling field is that of *co-evolution*. Husbands et al. [57] introduced a co-evolutionary distributed GA for tackling integrated manufacturing and scheduling problems. They consider a highly generalised version of the JSSP in which manufacturing plans for individual components are solved in parallel, taking into account the numerous interactions between them resulting from the shared use of resources. They utilise a multiple-species ecosystems model in which each population represents a feasible process plan for a particular component to be manufactured in a machine-shop. Separate populations evolve under the pressure of selection to find near-optimal process plans for each component. Their fitness functions however take into account the use of shared resources in the common machine-shop world. Thus, without the need for an explicit scheduling stage, a low-cost machine schedule emerges at the same time as the planes are being optimised. One population contains genotypes representing *arbitrators*—these co-evolve alongside the plans and resolve conflicts between the plans for differing components. Husbands et al. test their approach on 100 problems generated from data provided in Palmer [97] and give results for various criteria averaged over all the problems. The co-evolutionary GA is compared to simulated annealing and local dispatching rule heuristics and was found to outperform both of them. In another example, Jensen [63], two GAs evolve together in order to generate flexible schedules which can be adapted quickly after unforeseen breakdowns. One GA evolves patterns of possible breakdowns, rewarding those patterns which most schedules find difficult to adapt to. The other evolves schedules, rewarding those whose worst-case performance, when faced with a sample of breakdowns from the other GA, is the best. The results were good when tried on a small set of benchmark job-shop problems, but further research is needed.

Most recently, John in [65] present a co-evolutionary genetic algorithm for solving the benchmark scheduling instances given in [79], using makespan as the objective criterion. Bierwith and Mattfeld [10] introduced a hybrid scheduling algorithm which provided a mechanism for mixing non-delay and active schedulers using a fixed parameter δ. John [65] investigated the idea of co-evolving the parameter δ alongside operation sequences and found good results across the benchmark set. However, they report that this method still does not escape the problems generally associated with solving very large problems. A similar idea of co-evolving parameters was used by Kaschel et al. [67] who describe a GA for tackling a real-world shop floor scheduling problem for an engineering company using parallelization and parameter co-evolution. Finally, another real-world application of a co-evolutionary EA to a scheduling problem is given by Aickelin and Dowsland [1] which considers the problem of nurse-scheduling.

6.4. Hyper heuristics

Many of the state-of-the-art approaches to solving scheduling problems that have been described are extremely problem-specific and knowledge-intensive. Whilst providing good results on the particular problems for which they have been designed, they are expensive to implement in terms of development time and running-time of the algorithms. An emerging search technology which aims to circumvent these problems is that of *hyper-heuristics*—this term can be loosely defined as the process of using meta-heuristics to *choose* meta-heuristics to solve a problem. The key idea is to use a set of known, easy to implement and reasonably understood heuristics to transform the state of a problem; thus individual heuristics are associated with problem conditions under which the heuristic has been shown to flourish, and different heuristics are applied at different phases of the solution process. An early example of this idea to scheduling problems was given by Fang et al. [41], in which the hyper-heuristic approach was applied to open-shop scheduling problems. Fang et al. used an EA which built a solution in the following manner: A chromosome consisted of a series of pairs of integers $[t_0 - h_0, t_1 - h_1, \ldots, t_n - h_n]$ interpreted from left to right meaning, for each i, 'consider the t_ith uncompleted job and use heuristic h_i to select a task to insert into the schedule in the earliest possible place where it will fit'. A range of standard dispatch rules were used in the heuristic set. This approach produced excellent results on a set of benchmark problems, however this process still evolves solutions to individual problems rather than creating a generally applicable algorithm. A similar approach to solving job-shop problems was first suggested by Dorndorf and Pesch [33] and later improved by Hart and Ross [50]. Dorndorf's algorithm utilised Giffler and Thompson's algorithm to generate active schedules; chromosomes in their EA were of the form $(p_1, p_2, \ldots, p_{j*m})$, where each gene p_i encoded the priority dispatch rule (as described in Section 2) to be used when resolving the conflicts produced at the i_{th} iteration of the G&T algorithm. The algorithm was tested on a number of static instances of JSSP benchmarks, using makespan as the objective, and obtained results which although promising at the time have now been superseded by alternative methods, e.g. [116]. HGA, proposed by Hart et al. and already discussed in Section 2.1 is another example of a hyper-heuristic GA. Furthermore, the authors also observed that when constructing a schedule it is only the *early* choices that really matter—for example, if the first 50% of choices are made according to the evolved sequences, and then the remaining 50% made totally at random, then the results are still satisfactory. Although this could be an artefact of the problems themselves (and more research is required to determine this), this suggests a straightforward solution to some of the issues that can be raised concerning the use of EAs with very long chromosomes and the resulting danger of genetic drift, and also suggests that an EA could be used on very large scheduling problems. Cowling et al. [24, 48] use a genetic algorithm to evolve a sequence of calls to a set of low-level heuristics in order to solve a trainer scheduling problem. The length of the chromosome in this case is allowed to adapt, and longer chromosomes are penalised on the basis that they take longer to evaluate. Again in this work, the concept of the domain barrier remains, and the GA itself has no knowledge of the problem itself—it simply tries to evolve a good sequence of calls to the low-level problem specific heuristics.

7. Other evolutionary approaches

Aside from genetic algorithms, a variety of techniques based on evolutionary principles have been applied to scheduling problems. A selection of the more common ones are now reviewed.

7.1. GP

A small number of papers describe the use of genetic programming to tackle both benchmark and real-world scheduling problems. These papers tend to have a common theme, in that GP is used to evolve a heuristic or *dispatch rule* which is then used to schedule processes. For example, Dimopoulos and Zalzala [29–31] describe the use of GP to solve the single machine scheduling problem, in which GP used to evolve scheduling policies in the form of dispatching rules. These rules are trained to cope with different levels of tardiness and tightness of due dates. In a similar manner, Miyashita [82] also uses GP to generate dispatch rules, this time for a benchmark suite of job-shop scheduling problems, in which each problem had 10 jobs to be scheduled on 5 machines, and problems were characterized by 2 parameters, one governing distribution of job due-dates and release dates, and the other controlling the number of bottleneck resources. Miyashita compared the performance of GP to the Earliest Due-Date heuristic (EDD) and Shortest Processing Time (SPT) and found that the heuristic evolved by GP outperformed both of these rules. Atlan et al. [3] propose a general system to infer symbolic policy functions for distributed reactive scheduling in non-stationary environments. The job shop problem is used as a validating case study. Their system is based both on an original distributed scheduling model and on genetic programming for the inference of symbolic policy functions. The purpose is to determine heuristic policies that are local in time, long term near-optimal, and robust with respect to perturbations. Furthermore, the policies are local in state space: the global decision problem is split into as many decision problems as there are agents, i.e. machines in the job shop problem. In a real-world application [72], apply GP to scheudling maintenance of the national electricity grid in the UK. Maintenance must be planned so as to minimize costs taking into account: (1) location and size of demand, (2) generator capacities and availabilities, (3) electricity carrying capacity of the remainder of the network, that part not undergoing maintenance. Previous work showed the combination of a Genetic Algorithm using an order or permutation chromosome combined with hand coded "greedy" optimizers could readily produce an optimal schedule for a four node test problem and for the South Wales region of the UK high voltage power network. Langdon [72] describe the evolution of the best known schedule for the base South Wales problem using Genetic Programming starting from the hand coded heuristics used with the GA. Another real-world application of GP is described by Grimes [47], for scheduling maintenance of railway tracks. Track maintenance work was planned using both GA and GP, with profit as the optimisation criteria. The results where compared with an existing determinstic technique and it was found that the GP method gave the best results, with the GA method giving good results for short sections of track (10 miles) and poor results for long sections (50 miles).

Padman and Roehrig [96], apply GP to a resource-constrained project scheduling problem (RCPSP) with cash flows. Many heuristics exists for the RCPSP, but it has proven difficult to decide in advance which heuristic will provide the best result, given a problem characterization in terms of parameters such as size and complexity. Padman et al. discuss the use of genetic programming for heuristic selection, and compare it directly to alternative methods such as OLS regression and neural networks. The study indicates that the GP approach yields results that are an improvement on earlier methods. The GP solution also gives valuable information about project environments where a given heuristic is inappropriate.

7.2. Immune systems

The field of Artifical Immune Systems, or AIS, is rapidly emerging as a new research area in the general area of natural computation. The interested reader can refer to [27] for a detailed overview of the area in general, this section summarises the applications of the technique to scheduling problems. Mori et al. [45, 85, 86] proposes an immune algorithm to solve a scheduling problem which involved finding optimal batch-sizes for a number of jobs, then sequencing the processing of each batch on a number of machines, in order to optimise an objective function. Antibodies encoded batch-sizes and job priority, and their approach was inspired by the somatic and network theories of the immune system. Immune ideas relating to somatic recombination and hyper-mutation were used to maintain diversity within the antibody repertoire, whilst immune network ideas controlled the proliferation of antibodies within the system. King et al. [69] describe an agent based system based on the immune system for performing scheduling in computer systems. In this model, antigens perform recognition of specific hardware and/or software followed by an allocation response that will better suit the computer's resources to performing on-going and planned tasks. The system undergoes an evolutionary adaptation process, and also incorporates a memory. Cui et al. [25] apply a version of Mori's algorithm to multi-objective flowshop scheduling problems and argue that the immune approach is an important factor in preserving population diversity. They report results on 2-objective flowshop problems. Costa et al. [23] introduce an immune algorithm for make-span minimisation on parallel processors based on mimicking the cloning and proliferation aspects of the immune system. The algorithm was tested on 390 instances of generated problems in which each instance had i processors on which to schedule j jobs, with the processing time of each job obeying a uniform distribution in the range $[1, k]$. They compared their results to tabu search, simulated annealing, local search, and a number of heuristics and found that the immune based algorithm was effective in dealing with instances characterised by jobs with long processing times on a small number of machines, especially when compared to single-solution strategies. It is claimed that these superior results are due to the ability of the immune system approach to maintain diversity within a population of candidate solutions. Hart and Ross [51] and Hart et al. [53] investigate whether an artificial immune system can evolve robust schedules, i.e. schedules which cover a range of possible contingencies, both foreseeable and unforeseen. The authors utilise a genetic algorithm to evolve an immune system consisting of antibodies representing fragments of schedules, the idea being that that new schedules could rapidly be produced by combining these fragments. The antibody set was subjected to many different scheduling scenarios during its evolution, with the result that robust fragments that could be used in many different schedules survived. The issue of robustness is an important point and one that we return to later in this paper—this issue has often been ignored in evolutionary scheduling research, where the emphasis tends to be on producing schedules which optimize some particular criterion. In practice, an optimal schedule is not necessarily desired, especially if it is so fragile that a slight variation in conditions leads to collapse of the schedule. Practioners would tend to argue that 'good enough, fast enough' is a more sensible maxim to adopt, particularly if this leads to robust schedules which provide some leeway to accommodate small perturbations in environmental conditions,

7.3. Ant algorithms

A growing body of work reports on the use of ant algorithms for solving a variety of scheduling problems. The ant colony optimisation (ACO) meta-heuristic, described in for example, Corne et al. [21], is inspired by the principles of behaviour found in real ant colonies, has been applied to many NP-hard problems with promising results. In terms of scheduling, the results are rather mixed. The first application on ACO algorithm to the JSSP was reported in [19], although this approach obtained relatively poor computational results. Further work on the job-shop problem was reported by Dorigo et al. [32]. In this work, problems were represented as undirected weighted graphs and the ACO approach applied to JSP instances of dimension 10×10 and 10×15. A result within 10% of the known optimum value was obtained in both cases. Van der Zwann et al. applied a similar method to the benchmark problems $6/6/G/C_{\max}$, $10/10/G/C_{\max}$, (due to Muth-Thompson) and the "la26" $20/10/G/C_{\max}$ problem due to Lawrence. Although the optimum value was obtained for the $6/6/G/C_{\max}$ problem, the best value obtained on $10/10/G/C_{\max}$ was within 8% of the best known optimum and on $20/10/G/C_{\max}$ within 26% of the best known optimum. However these results are relatively poor compared to other published EA approaches—Yamada and Nakano [88] obtained the optimum result on the $10/10/G/C_{\max}$ problem using an EA approach in 1992. Thus, on JSSPs, ant algorithms have yet to prove competitive with the current state-of-the-art algorithms. However, Stützle [110] applied a version of an ACO known as $\mathcal{MIN}-\mathcal{MAX}$ [111] to the flowshop problem, and obtained promising results that are better than or at least comparable to other state-of-the-art algorithms. In this work, the ACO approach was hybridised with a local-search procedure based on an insertion neighbourhood, in which a new permutation is obtained by removing the job at position i and inserting it at position j. The Single Machine Total Weighted Tardiness Problem, SMTWTP has been tackled in [28] using an ACO algorithm again combined with a powerful local search algorithm, known ACS-SMTWTP. This algorithm was tested on a benchmark set of 100 instances of SMTWTPs available from the OR-library and could find the optimal or best-known solution on all instances of the test set in a single run. It was compared to a number of other algorithms, and the authors found that only the iterated dynasearch algorithm of Congram et al. [20] was comparable. ACO algorithms have also been applied to the unweighted version of this problem, the Single Machine Total Tardiness Problem, for example by Bauer et al. [6], whose results were later improved upon by Merkle and Middendorf [80]. Finally, Merkle et al. [81] have applied an ACO to the Resource Constrained Scheduling Problem, RCSP. The authors studied the performance of their algorithm on a test set of problems obtained from the Project Scheduling Library, PSPLIB [71] and compared it to the best known results at the time on these problems (described in [55], including genetic algorithms, simulated annealing and other randomized heuristics) and found that the ACO algorithm performed best under the constraint that every algorithm was allowed to compute and evaluated the same restricted number of solutions.

8. Bridging the gap

Recently, there has been a growing trend of workshops addressing the significant point of how evolutionary scheduling ideas can bridge the gap from the academic to the practical more successfully, for example see [74]. This section addresses some of the problems evolutionary scheduling currently faces and suggests directions for future research.

8.1. Reconfigurable schedulers

The major stumbling block in producing commercially viable schedulers to be utilised for real in industry appears to be cost, witness for example the account by Ottner [95] describing real-life implementation of a genetic scheduler for scheduling commercial advert breaks for a British television channel. Montana describes the case for a ’reconfigurable’ scheduler, in [84], which can handle a variety of scheduling applications and domains without modification of its software. Having such a scheduler would then provide a means for performing low-cost and therefore practical scheduling. Certainly such systems exist in the non-evolutionary world, for example Fourer et al. describe a language AMPL which can be used to express scheduling problems. Once expressed, they can be solved by multiple solvers, including one from ILOG. Montana et al. introduce an open-source system called Vishnu in the above paper which provides a framework in which to define problems and a genetic scheduler wrapped into a web-based architecture. Clearly, such a scheduler cannot provide optimal performance across all classes of problems, but often, the solution quality will suffice for practical purposes. As discussed in Section 6.4, the hyper-heuristic approach appears promising in this respect—the hyper-heuristic framework provides a generic scheduling engine into which problem specific operators can be plugged.

8.2. Robustness

In Section 7.2 the issue of *robustness* in schedules was referred to. This is an important point—optimal schedules are seldom required in an industrial environment, rather something that is resilient to the inevitable environmental changes that occur. This issue has been addressed in some evolutionary work: Wu et al. [124] tackles the question of making schedules robust to disturbances in operation processing times, but does not consider other possible events that may perturb the original schedule. This work employed a graph-theoretic approach combined with a branch-and-bound algorithm. Herrmann [56] used a co-evolutionary algorithm to tackle a similar problem in which the algorithm converges to the worst case and therefore most robust scenario. Jensen and Hansen [64] propose a new method of creating robust solutions to job-shop problems that produces solutions which are robust to breakdowns of machines, and as previously mentioned, Hart and Ross [51] and Hart et al. [53] have adopted an immune system approach to tackle the issue of disturbances in arrival-dates. All of these papers contain promising ideas, however the issue of robustness is one that needs to be adopted more often by the evolutionary scheduling community if evolutionary algorithms are to be a serious contender for solving practical scheduling problems.

8.3. Scalability/large problems

A further hurdle regarding application of evolutionary scheduling techniques to practical problems lies with the size of real problems—so far, research has concentrated on relatively small problems, even when dealing with real-world cases, with only a few exceptions In contrast, many real problems drawn from the manufacturing, construction, and distribution sectors can involve several thousands of tasks. For example, yearly shutdown for reloading and maintenance of a nuclear power installation requires about 5,000 tasks to be scheduled in about a month, while the production and distribution of a national magazine or newspaper with regional variations can involve up to 20,000 weekly or monthly tasks. Good solutions to such problems can mean massive savings in time and cost, but it is unclear whether the range of techniques successfully developed for smaller problems will suitably scale up to larger cases. Techniques from other domains have been successfully applied to such large problems, therefore the challenge for evolutionary scheduling is to be able to replicate this success. For example, an interesting area in which large problems predominate is in planning and scheduling for space. Chien et al. [17] report an iterative constructive heuristic method for antenna tracking operations scheduling in NASA's Deep Space Network (DSN); the DSN is a network of complexes and antennas whose functions include to receive telemetry signals from spacecraft, transmit spacecraft control commands, and generate the radio navigation data used to locate and guide spacecraft. Optimised planning and scheduling of DSN's operations is at a premium, especially with the planned New Millennium program [89]. Currently, the DSN performs scores of antenna tracks to support earth orbiting and deep space missions. Again, the optimal scheduling of resources and sequencing for these operations is an example of a daily problem with complex constraints and many hundreds of tasks. Evolutionary techniques, with a commonly mooted strength at being able to optimise well over complex objectives and criteria, would seem to have much to offer in the area of large problems, but the community needs to deal with the inevitable time complexity issues involved. There do appear to be ways forward, however—Rana and Whitley's [101] describe the speedup gained through a coarse-grained simulation approach, and a range of hybrid or hierarchical approaches find better results through using an evolutionary algorithm together with a local search or constructive method, but without undue extra time complexity.

8.4. Comparative work

There is no doubt that evolutionary techniques have merit in general when applied to many kinds of scheduling problems. Much existing work reports fair competition between evolutionary approaches and other methods, finding justification for the superiority of the evolutionary method or methods tried [35, 39, 68, 125]. But, there is also justification for fair criticism of much other work in the field which fails to perform such comparative study. There are several good underlying reasons for this lack of comparison. For example, reported work may indicate that preliminary experiments with simpler methods have found the evolutionary technique to seem better, and so the work reported in the article centres on the 'later' investigations. In related cases, researchers may report on a successful GA/local search hybrid, and somewhere say (though often not present the experimental details) that the hybrid was found better and/or more robust than either method alone. Commonly, however, there *seems* to be a tendency to *a priori* expect the evolutionary algorithm to be worth pursuing above less sophisticated techniques such as stochastic hillclimbing or simulated annealing. Often this is not justified, as was found by Juels and Wattenberg [66]. Juels and Wattenberg compared a simple stochastic hillclimbing approach to a previously published GA approach to job shop scheduling problems, and found the results to be competitive, and much faster. Also interesting, however, is that they went on to define an alternative representation strategy, and this time found the genetic algorithm to defeat the hillclimber. In several cases though, it is certainly not clear whether or not a more simple approach would yield better results, or perhaps occupy a more attractive region of the speed/quality tradeoff curve. Perhaps more importantly, non-comparison slows down a key line of needed research effort, which is to properly assess and determine the 'niche' for evolutionary algorithms work in scheduling. Where comprehensive comparative study has been done, strong pointers to such 'niches' have been found. For example, Rana [101] points to a niche for GAs in fast, stochastic, and large real-world scheduling problems. Fang et al. [39] points towards a general niche for GA's in larger scheduling problems with quality criteria which are notoriously awkward for heuristic approaches, such as just-in-time oriented measures.

8.5. Benchmark data and problem generators

There are several sources of benchmark data for scheduling problems, and this is essential if a more thorough approach to comparative work is to be undertaken. A list of commonly used sources is given in Table 5, and an article by Drummond [34] provides information on a number of benchmark problem sets. However, assessing performance on sets of benchmarks

Table 5. Sources of benchmark scheduling data.

Type of problem	Source
Flowshop, jobshop, openshop, crew scheduling	[7]
Flowshop, jobshop, openshop (including source code)	[114]
Jobshop	[87]
Small jobshop	[42]
Resource Constrained Project Scheduling	[90]
NASA payload processing scheduling	[77]

still leaves room for some gross over-generalisation, i.e that the observed performance of an EA on a small set of problems can be extrapolated to whole classes of other problems. In order to determine the niche(s) in which evolutionary scheduling algorithms might potentially outperform other algorithms, a more rigorous approach is required. Before one can claim success in applying a particular algorithm, one should show that it works over a wide range of problems, with the problems chosen such that they exhibit a variety of features and vary in difficulty in some configurable way. Rather than use benchmarks problems then, it is more useful to use a parameterised problem generator which can generate problem instances at random, but in some tunable manner, so that many different instances of problem classes can be generated. For instance, this approach was adopted by Ross et al. [103] who applied an EA to a timetabling problem and showed unexpectedly that for some classes of problems, although *highly* constrained problems and *lightly* constrained problems were easy to solve, problems of 'medium' difficulty were extremely hard for the EA. This type of approach has been adopted in evolutionary scheduling work by Hart and Ross [52] and Watson et al. [119]. Hart and Ross [52] tackle the problem of generating tunable jobshop problems. In their generator, problems are produced in which one can specify a variety of features: the number of tasks to be scheduled and the number of machines can be set; the total amount of idle time allowed per machine can be chosen; the amount of slack in the arrival and due-dates of each job can also be specified. Distributions specifying the processing times of each task can be chosen to be either Gaussian or uniform, and the means and standard deviations chosen in each case. In the latter work, Watson et al. describe a generator which produces *job-correlated* and *machine correlated* flowshop scheduling problems with non-random structure. In machine-correlated problems, the processing times of all jobs on a specific machine are sampled from a relatively tight distribution relative to that machine, whereas in job-correlated problems, processing times are correlated in that they are sampled from a relatively tight distribution relative to the particular job. Processing times are drawn from a number of Gaussian distributions in each case. The generator is tunable in that bounds on the distribution standard deviations, distribution means and distribution overlaps can be set. Future work in this direction appears key to presenting a clear case for evolutionary scheduling.

9. Conclusion

In conclusion, evolutionary scheduling has come a long way since the poineering days of applying GAs with binary representations to academic problems. Algorithms now exist that are capable of tackling large and hard real-world problems, and that are competitive with more traditional techniques. Attention to some of the areas identified in Section 8 will further help in proving the case for the addition of evolutionary algorithms to the toolkit of anyone seriously interested in solving difficult real-world problems, however, the evidence to date surely suggests that such work will be worthwhile, given the solid foundations on which the field now rests.

Acknowledgments

The first two authors are grateful for support from the UK EPSRC, under research grant number GR/N36660.

References

1. U. Aickelin and K. Dowsland, “Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem,” Journal of Scheduling, vol. 3, no. 3, pp. 139–153, 2000.
2. I. Al-Harkan, “On merging sequencing and scheduling theory with genetic algorithms to solve stochastic job shops,” PhD thesis, University of Oklahoma, 1997.
3. L. Atlan, J. Bonnet, and M. Naillon, “Learning distributed reactive strategies by genetic programming for the general job shop problem,” in Proceedings of the 7th annual Florida Artificial Intelligence Research Symposium, D. Dankel and J. Stewman (Eds.), IEEE Press, Pensacola, Florida, USA, May 1994.
4. T. Bagchi, Multiobjective Scheduling by Genetic Algorithms, Kluwer: Boston, 1999.
5. S. Bagchi, S. Uckun, Y. Miyabe, and K. Kawamura, “Exploring problem-specific recombination operators for job shop scheduling,” in Proceedings of the Fourth International Conference on Genetic Algorithms, R. Belew and L. Booker (Eds.), Morgan Kaufmann: San Mateo, 1991, pp. 10–17.
6. A. Bauer, B. Bullnheimer, R. Hartl, and C. Strauss, “An ant colony optimization approach for the single machine tardiness problem,” in Proceedings of the 1999 Congress on Evolutionary Computation, P. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala (Eds.), IEEE Press, 1999, pp. 1445–1450.
7. J. Beasley, “Or-library: Distributing test problems by electronic mail,” Journal of the Operational Research Society, vol. 41, no. 11, pp. 1069–1072, 1990.
8. R. Belew and B. L. Booker (Eds.), in Proceedings of the Fifth International Conference on Genetic Algorithms. Morgan Kaufmann: San Mateo, 1991.
9. C. Bierwirth, D. Mattfeld, and H. Kopfer, “On permutation representations for scheduling problems,” in Parallel Problem Solving from Nature: PPSN IV, Y. Davidor, H.-P. Schwefel, and R. Manner (Eds.), Springer-Verlag: Berlin, 1996, LNCS 1141, pp. 310–318.
10. C. Bierwith and D. Mattfeld, “Production scheduling and rescheduling with genetic algorithms,” Evolutionary Computation, vol. 7, no. 1, pp. 1–17, 1999.
11. J. Blackstone, D. Phillips, and G. Hogg, “A state-of-the-art survey of dispatching rules for manufacturing job operations,” International Journal of Production Research, vol. 20, pp. 27–45, 1982.
12. C. Brizuela and R. Aceves, “Experimental genetic operators analysis for the multi-objective permutation flowshop,” in Evolutionary Multicriterion Optimization; EMO 2003, C. Fonseca, P. Fleming, E. Zitzler, K. Deb, and L. Thiele (Eds.), Springer-Verlag: Berlin, 2003, pp. 578–592.
13. R. Bruns, “Direct chromosome representation and advanced genetic algorithms for production scheduling,” in Proceedings of the Fifth International Conference on Genetic Algorithms, S. Forrest (Ed.), Morgan Kaufmann: San Mateo, Feb. 1993, pp. 352–359.
14. A. Cardon, T. Galinho, and J.-P. Vacher, “A multi-objective genetic algorithm in job shop scheduling problem to refine an agents’ architecture,” in Proceedings of EUROGEN’99, K. Miettinen, M. M. Mäkelä, P. Neittaanmäki, and J. Periaux (Eds.), University of Jyváskylä, Jyväskylä, Finland, 1999.
15. H. M. Cartwright and A. L. Tuson, “Genetic algorithms and flowshop scheduling: Towards the development of a real-time process control system,” in Selected Papers: AISB Workshop on Evolutionary Computing, T. C. Fogarty (Ed.), Lecture Notes in Computer Science No. 865, Springer Verlag, 1994, pp. 277–290.
16. R. Cheng, M. Gen, and Y. Tsujimura, “A tutorial survey of job-shop scheduling problems using genetic algorithms—i. representation,” Computers and Industrial Engineering, vol. 30, no. 4, pp. 983–997, 1996.
17. S. Chien, A. Govindjee, T. Estlin, X. Wang, T. Fisher, and R. H. Jr, “Automating generation of tracking plans for a network of communications antennas,” in International Workshop on Planning and Scheduling for Space Exploration and Science: Workshop Notes, S. Chien (Ed.), NASA JPL, 1997.
18. G. A. Cleveland and S. F. Smith, “Using genetic algorithms to schedule flow shop releases,” in Proceedings of the Third International Conference on Genetic Algorithms and their Applications, J. D. Schaffer (Ed.), Morgan Kaufmann: San Mateo, 1989, pp. 160–169.
19. A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian, “Ant system for job-shop scheduling,” JORBEL—Belgian Journal of Operations Research, Statistics and Computer Science, vol. 34, pp. 39–53, 1994.
20. R. Congram, C. Potts, and S. Van de Velde, “An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem,” Technical report, Faculty of Mathematical Studies, University of Southhampton, 1998.
21. D. Corne, M. Dorigo, and F. Glover (Eds.), in New Ideas in Optimization, chap. Ant Colony Optimization. McGraw-Hill: London, 1999.

22. D. Corne and J. Ogden, “Evolutionary optimisation of methodist preacher timetables,” in PATAT 97: Practice and Theory of Automated Timetabling II, 1997, pp. 142–156.
23. A. Costa, P. Vargas, F. Von Zuben, and P. Franca, “Makespan minimisation on parallel processors: An immune based approach,” in Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002), Fogel et al. (Eds.), IEEE Press, 2002, pp. 920–926.
24. P. Cowling, G. Kendal, and L. Han, “An investigation of a hyper-heuristic genetic algorithm applied to a trainer scheduling problem,” in Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002), Fogel et al. (Eds.), IEEE Press, 2002, pp. 118–1190.
25. X. Cui, M. Li, and T. Fang, “Study of population diversity of multiobjective evolutionary algorithm based on immune and entropy principles,” in Proceedings of the IEEE Congress on Evolutionary Computation, IEEE Press: Piscataway, NJ, 2001, pp. 1316–1321.
26. L. Davis, “Job shop scheduling with genetic algorithms,” in Proceedings of the International Conference on Genetic Algorithms and their Applications, J. J. Grefenstette (Ed.), Morgan Kaufmann: San Mateo, 1985, pp. 136–140.
27. L. de Castro and J. Timmis, Aritifical Immune Systems: A New Computational Intelligence Paradigm. Springer, London, 2002.
28. M. den Besten, T. Stützle, and M. Dorigo, “Ant colony optimization for the total weighted tardiness problem,” in Parallel Problem Solving from Nature: 6th International Conferencence, M. Schoenauer et al. (Eds.), Number 1917 in Lecture Notes in Computer Science, Springer Verlag, 2000, pp. 611–620.
29. C. Dimopoulos and A. M. S. Zalzala, “Evolving scheduling policies through a genetic programming framework,” in Proceedings of the Genetic and Evolutionary Computation Conference, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith (Eds.), Morgan Kaufmann: Orlando, Florida, USA, 13–17 July 1999, vol. 2, pp. 1231.
30. C. Dimopoulos and A. M. S. Zalzala, “A genetic programming heuristic for the one-machine total tardiness problem, “in Proceedings of the Congress on Evolutionary Computation, P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala (Eds.), IEEE Press: Mayflower Hotel, Washington, D.C., USA, 6–9 July 1999, vol. 3, pp. 2207–2214.
31. C. Dimopoulos and A. M. S. Zalzala, “Investigating the use of genetic programming for a classic one-machine scheduling problem,” Advances in Engineering Software, vol. 32, pp. 489–498, 2001.
32. M. Dorigo, V. Maniezzo, and A. Colorni, “The ant system: Optimization by a colony of cooperating anrts,” IEEE Trans. Systems, Man and Cybernetics—Part B, vol. 26, pp. 29–41, 1996.
33. U. Dorndorf and E. Pesch, “Evolution based learning in a job shop scheduling environment,” Computers in Operations Research, vol. 22, pp. 25–40, 1995.
34. M. Drummond, “Scheduling benchmarks and related resources,” Newsletter of the AAAI SIGMAN, vol. 6, no. 3, 1993.
35. F. Easton and N. Mansour, “A distributed ga for employee staffing and scheduling,” in [**?**], 1993, pp. 360–367.
36. A. Eiben and C. Schippers, “Multi-parent’s niche: n-ary crossovers on nk-landscapes,” in Proceedings of the 4th Conference on Parallel Problem Solving from Nature, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel (Eds.), 1996, pp. 319–328.
37. S. Esquivel, G. Leguizamon, F. Zuppa, and R. Gallard, “A performance comparison of alternative heuristics for fsp,” in Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2002, S. Cagnoni et al. (Eds.), Springer, Berlin, 2002, pp. 51–60.
38. H.-L. Fang, “Genetic algorithms in timetabling and scheduling,” PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1994.
39. H.-L. Fang, D. Corne, and P. Ross, “A genetic algorithm for job-shop problems with various schedule quality criteria,” in Evolutionary Computing: 1996 AISB Workshop: Selected Papers, T. Fogarty (Ed.), Lecture Notes in Computer Science 1143, Springer, 1996, pp. 39–49.
40. H.-L. Fang, P. Ross, and D. 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, S. Forrest (Ed.), Morgan Kaufmann: San Mateo, 1993, pp. 375–382.
41. H.-L. Fang, P. Ross, and D. Corne, “A promising hybrid ga/heuristic approach to open-shop scheduling problems,” in ECAI’94: Proceedings of the 11th European Conference on Artificial Intelligence, A. Cohn (Ed.), Wiley, 1994, pp. 590–594.

42. H. Fisher and G. L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," in Industrial Scheduling, J. F. Muth and G. L. Thompson (Eds.), Prentice Hall: Englewood Cliffs, New Jersey, 1963, pp. 225–251.
43. D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton (Eds.), in Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002). IEEE Press, 2002.
44. S. French, Sequencing and Scheduling, John Wiley: New York, 1982.
45. T. Fukuda, M. Mori, and M. Tsukiyama, "Immune networks using genetic algorithms for adaptive production scheduling," in Proceedings of the 15th IFAC World Congress, G. Goodwin and R. Evans (Eds.), Pergamon Press Ltd.: London, pp. 57–60.
46. B. Giffler and G. Thompson, "Algorithm for solving production scheduling problems," Operations Research, vol. 8, no. 4, pp. 487–503, 1960.
47. C. A. Grimes, "Application of genetic techniques to the planning of railway track maintenance work," in First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA, A. M. S. Zalzala (Ed.), IEE: Sheffield, UK, 12–14 Sept. 1995, vol. 414, pp. 467–472.
48. L. Han, G. Kendall, and P. Cowling, "An adaptive length chromosome hyperheuristic genetic algorithm for a trainer scheduling proble," Technical Report NOTTCS-TR-2002-5, University of Nottingham, 2002.
49. M. Hapke, A. Jaskiewicz, and K. Kurowski, "Multi-objective genetic local search methods for the flowshop-problem," in Advances in Nature-Inspired Computation: The PPSN VII Workshops, PEDAL, University of Reading, UK, 2002, pp. 22–23.
50. E. Hart and P. Ross, "A heuristic combination method for solving job-shop scheduling problems," in Parallel Problem Solving from Nature—PPSN V, A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel (Eds.), Springer: Berlin, 1998, pp. 845–854, Lecture Notes in Computer Science 1498.
51. E. Hart and P. Ross, "An immune system approach to scheduling in changing environments," in Genetic and Evolutionary Computation Conference—GECCO 1999, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith (Eds.), 1999, pp. 1559–1565.
52. E. Hart and P. Ross, "A systematic investigation of ga performance on jobshop scheduling problems," in Real-World Applications of Evolutionary Computing, S. Cagnoni et al. (Eds.), Springer, 2000, pp. 277–287.
53. E. Hart, P. Ross, and J. Nelson, "Producing robust schedules via an artificial immune system," in IEEE World Congress on Computational Intelligence, ICEC 1998, IEEE Press, 1998, pp. 464–469.
54. E. Hart, P. Ross, and J. Nelson, "Solving a real world problem using an evolving heuristically driven schedule builder," Evolutionary Computation, vol. 6, no. 1, pp. 61–80, 1998.
55. S. Hartmann and R. Kolisch, "Experimental evaluation of state-of-the-art heuristics for the resource-constrainted scheduling problem," European Journal of Operations Research, vol. 127, no. 1, pp. 394–407, 2000.
56. J. Herrmann, "A genetic algorithm for minimax optimzation problems," in Proceedings of the 1999 Congress on Evolutionary Computation, P. Angeline, Z. Michalewicz, M. Schoenhauer, X. Yao, and A. Zalzala (Eds.), IEEE Press, 1999, vol. 2, pp. 1099–1103.
57. P. Husbands, M. McIlhagga, and R. Ives, Handbook of Evolutionary Computation, chap. 'An ecosystem model for integrated production planning,' Institute of Physics Publishing and Oxford University Press: Bristol, New York, 1997, pp. G9.3:1–8.
58. P. Husbands and F. Mill, "Simulated co-evolution as the mechanism for emergent planning and scheduling," in Proceedings of the Fifth International Conference on Genetic Algorithms, 1991, R. Belew and B. L. Booker (Eds.), Morgan Kaufmann: San Mateo, 1991, pp. 264–270.
59. P. Husbands, F. Mill, and S. Warrington, "Genetic algorithms, production planning optimisation and scheduling," in Parallel Problem Solving from Nature, H.-P. Schwefel and R. Männer (Eds.), vol. 496 of Lecture Notes in Computer Science, Springer, 1990, pp. 80–84.
60. H. Ishibuchi and T. Murata, "Multi-objective genetic local search algorithm and its application to flowshop scheduling," IEEE Transactions on Systems, Man and Cybernetics, vol. 28, no. 3, pp. 392–403, 1998.
61. A. Jain and S. Meeran, "A state-of-the-art review of job-shop scheduling techniques," Technical report, University of Dundee, 1998.
62. A. Jan, M. Yamamoto, and A. Ohuchi, "Evolutionary algorithms for nurse scheduling problem," in Congress on Evolutionary Computation, 2000, IEEE Press: Piscataway, NJ, pp. 196–203.

63. M. Jensen, "Finding worst-cae flexible schedules using co-evolution," in Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001, Spector et al. (Eds.), Morgan Kaufmann: San Francisco, California, USA, 2001, pp. 1144–1151.
64. M. Jensen and T. Hansen, "Robust solutions to job-shop problems," in Proceedings of the 1999 Congress on Evolutionary Computation, P. Angeline, Z. Michalewicz, M. Schoenhauer, X. Yao, and A. Zalzala (Eds.), IEEE Press, 1999, vol. 2, pp. 1138–1144.
65. D. J. John, "Co-evolution with the bierwith-mattfeld hybrid scheduler," in Proceedings of the Genetic and Evolutionary Computation Conference (GEECCO), W. B. Langdon et al. (Eds.), Morgan Kaufmann Publishers: New York, 2002, pp. 259.
66. A. Juels and M. Wattenberg, "Stochastic hillclimbing as a baseline method for evaluating genetic algorithms," Technical Report UCB Technical Report CSD-94-834, Department of Computer Science, University of California at Berkeley, 1994.
67. J. Kaschel, B. Meier, M. Fischer, and T. Teich, "Evolutionary real-world shop floor scheduling using parallelization and parameter coevolution," in Proceedings of the Genetic and Evolutionary Computation Conference, D. Whitley et al. (Eds.), Morgan Kaufmann: San Francisco, CA, 2000, pp. 697–701.
68. M. Kidwell, "Using genetic algorithms to schedule distributed tasks on a bus-based system," in Proceedings of the Fifth International Conference on Genetic Algorithms, S. Forrest (Ed.), Morgan Kaufmann: San Mateo, 1993, pp. 368–374.
69. R. King, S. Russ, A. Lambert, and D. Reese, "An artificial immune system model for intelligent agents," Future Generation Computer Systems, Elsevier Science, vol. 17, pp. 335–343, 1999.
70. S. Kobayashi, I. Ono, and M. Yammamura, "An efficient genetic algorithm for job shop scheduling problems," in ICGA, L. J. Eshelman (Ed.), Morgan Kaufmann, 1995, pp. 506–511.
71. R. Kolisch and A. Sprecher, "Psplib—a project scheduling problem library," Eurpoean Journal of Operations Research, vol. 96, no. 1, pp. 205–216, 1996.
72. W. Langdon, "Scheduling maintenance of electrical power transmission networks using genetic programming," in Artificial Intelligence Techniques in Power Systems, K. Warwick, A. Ekwue, and R. Aggarwal (Eds.), IEE Press, UK, 1997, chap. 10, pp. 220–237.
73. W. Langdon, "Scheduling planned maintenance of the south wales region of the national grid," in Evolutionary Computing: AISB International Workshop 1997: Selected Papers, D. Corne and J. Shapiro (Eds.), Lecture Notes in Computer Science 1305, Springer: Berlin, 1997, pp. 181–197.
74. W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska (Eds.) in GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann Publishers: New York, 2002.
75. S.-C. Lin, E. Goodman, and W. Punch, "Investigating parallel genetic algorithms on job shop scheduling problems," in Evolutionary Programming VI, P. Angeline, R. Reynolds, J. McDonnell, and R. Eberhart (Eds.), Lecture Notes in Computer Science 1213, Springer, 1997, pp. 383–393.
76. S.-C. Lin, E. D. Goodman, and W. F. Punch, "A genetic algorithm approach to dynamic job-shop scheduling problems," in Proceedings of the Seventh International Conference on Genetic Algorithms, T. Back (Ed.), Morgan-Kaufmann, 1997, pp. 481–489.
77. Mark drummond. http://fi-www.arc.nasa.gov/fia/projects/xfr/papers/benchmark-article.html.
78. S. Matsui, I. Watanabe, and K. Tokoro, "Real-coded parameter-free genetic algorithms for job-shop scheduling," in Parallel Problem Solving from Nature—PPSN VII, J. Merelo-Guervos, P. Adamidis, H.-G. Beyer, J.-L. Fernandez-Villacanas, and H.-P. Schwefel (Eds.), 2002, pp. 801–810.
79. D. C. Mattfield, Evolutionary Search and the Job-Shop, Physica-Verlag, Heidelberg, 1996.
80. D. Merkle and M. Middendorf, "An ant algorithm with a new pheromone evaluation rule for total tardiness problems," in Applications of Evolutionary Computing, EvoWorkshops (2000), S. Cagnoni et al. (Eds.), vol. 1803 of LNCS, Springer-Verlag, 2000, pp. 287–296.
81. D. Merkle, M. Middendorf, and H. Schmeck, "Ant colony optimization for resource-constrained project scheduling," in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), D. Whitley et al. (Eds.), Morgan Kaufmann, 2000: Las Vegas, Nevada, USA, July 8–12, 2000, pp. 893–900.
82. K. Miyashita, "Job-shop scheduling with GP," in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000), D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer (Eds.), Morgan Kaufmann: Las Vegas, Nevada, USA, 10–12 July 2000, pp. 505–512.

83. D. Montana, G. Bidwell, G. Vidaver, and J. Herrero, "Scheduling and route selection for military land moves using genetic algorithms," in Congress on Evolutionary Computation, P. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala (Eds.), IEEE Press, 1999, pp. 1118–1123.
84. D. Montanta, "A reconfigurable optimising scheduler," in Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001, L. Spector et al. (Eds.), Morgan Kaufmann: San Francisco, California, USA, 2001, pp. 1159–1166.
85. M. Mori and T. Fukuda, "Immune algorithm and its application to factory load dispatching," in Proc. of the JAPAN-USA Symposium on Flexible Automation, 1994, pp. 1343–1346.
86. M. Mori, M. Tsukiyama, and T. Fukuda, "Adapative scheduling system inspired by the immune system," in Proceedings of the IEEE Conference on Systems, Man and Cybernetics, IEEE Computer Society Press: Los Alamitos, US, 1998, pp. 3833–3837.
87. T. Morton and D. Pentico, Heuristic Scheduling Systems, John Wiley: New York, 1993.
88. R. Nakano and T. Yamada, "Conventional genetic algorithm for job shop problems," in Proceedings of the Fifth International Conference on Genetic Algorithms, R. Belew and B. L. Booker (Eds.), Morgan Kaufmann: San Mateo, 1991.
89. Nasa new millenium program. http://nmp.jpl.nasa.gov/.
90. Neosoft. http://www.NeoSoft.com/ benchmarx/.
91. B. Norman and J. Bean, "Random keys genetic algorith m for job shop scheduling," Technical report, Department of Industrial and Operations Engineering, University of Michigan, 1994.
92. B. Norman and J. Bean, "Operation sequencing and tool assignment for multiple spindle cnc machines," in ICEC'97: Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, IEEE Press: Piscataway, NJ, 1997, pp. 425–429.
93. E. Nowicki and C. Smutnicki, "A fast tabu search algorithm for the permutation flowshop problem," European Journal of OR, Elsevier, vol. 91, pp. 160–175, 1996.
94. I. Ono, M. Yamamura, and S. Kobayashi, "A genetic algorithm for job shop scheduling problems using job-based order crossover," in ICEC'96: Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, IEEE, 1996, pp. 547–552.
95. S. Ottner, "Developing scheduling software using genetic algorithms in a commercial environment," in GECC) 2002: Proceedings of the Genetic and Evolutionary Computation Conference, W. B. Langdon et al. (Eds.), Morgan Kaufmann Publishers: New York, 2002, pp. 80–87.
96. R. Padman and S. Roehrig, "A genetic programming approach for heuristic selection in constrained project scheduling," in Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies, R. S. Barr, R. V. Helgason, and J. L. Kennington (Eds.), Kluwer Academic Publishers: Norwell, MA, USA, 1997, chap. 18, pp. 405–421.
97. G. Palmer, An Integrated Approach to Manufacturing Planning. PhD thesis, School of Engineering, University of Huddersfield, 1994.
98. S. Panwalker and W. Iskander, "A survey of scheduling rules," Operations Research, vol. 25, no. 1, pp. 45–61, 1977.
99. P. Poon and N. Carter, "J. Genetic algorithm crossover operators for ordering applications," Computers and Operations Research, vol. 22, pp. 135–147, 1995.
100. E. Ramat, G. Venturini, C. Lente, and M. Simane, "Solving the multiple resource constrained project scheduling problem with a hybrid genetic algorithm," in Proceedings of the Seventh International Conference on Genetic Algorithms, T. Back (Ed.), Morgan Kaufmann, 1997, pp. 489–496.
101. S. Rana, A. Howe, K. Mathias, and D. Whitley, "Comparing heuristic, evolutionary and local search approaches to scheduling," in The Third Artificial Intelligence Planning Systems Conference—AIPS-96, B. Drabble (Ed.), AAAI Press, 1996, pp. 174–181.
102. C. Reeves and T. Yamada, "Genetic algorithms, path-relinking and the flowshop sequencing problem," Evolutionary Computation, vol. 6, pp. 45–60, 1998.
103. P. Ross, E. Hart, and D. Corne, Some Observations about ga Based Timetabling, Springer-Verlag: Heidelberg, 1997, pp. 115–130.
104. K. R. Ryu, J. Hwang, H. R. Choi, and K. K. Cho, "A genetic algorithm hybrid for hierarchical reactive scheduling," in Proceedings of the Seventh International Conference on Genetic Algorithms, T. Back (Ed.), Morgan Kaufmann, 1997, pp. 497–505.

105. J. Sakuma and S. Kobayashi, "Extrapolation-directed crossover for job-shop scheduling problems: Complementary combination with jox," in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), D. Whitley et al. (Eds.), Morgan Kaufmann, 2000, Las Vegas, Nevada, USA, July 8–12, 2000, pp. 973–980.
106. H. Sawai and S. Kizu, "Parameter-free genetic algorithm inspired by "disparity theory of evolution,"" in Proceedings of PPSN-V, Springer-Verlag: Berlin-Heidelberg, 1998, pp. 702–711.
107. K. Shaw and P. Fleming, "Initial study of practical multi-objective genetic algorithms for scheduling the production of chilled ready meals," in Proceedings of Mendel'96, the 2nd International Mendel Conference on Genetic Algorithms, P. Osmera (Ed.), ISBN 80-214-0769-7, 1996.
108. L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke (Eds.), in Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001, Morgan Kaufmann: San Francisco, California, USA, 2001.
109. T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley, "A comparison of genetic sequencing operators," in Proceedings of the Fifth International Conference on Genetic Algorithms, R. Belew and B. L. Booker (Eds.), Morgan Kaufmann: San Mateo, 1991.
110. T. Stützle, "An ant approach for the flowshop problem," in Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing (EUFIT'98), H. Zimmerman (Ed.), Verlag Mainz: Aachen, Germany, 1998, vol. 3, pp. 1560–1564.
111. T. Stützle and H. Hoos, "Improvements on the ant system: Introducing $\mathcal{MAX}_{\mathcal{M}}\mathcal{IN}$ ant system," in Artificial Neural Networks and Genetic Algorithms, G. Smith and R. Steele (Eds.), Springer-Verlag: Wien, 1998, pp. 245–259.
112. G. Syswerda, "Schedule optimization using genetic algorithms," in Handbook of Genetic Algorithms, L. Davis (Ed.), New York: Van Nostrand Reinhold, 1991, pp. 332–349.
113. E. Taillard, "Some efficient heuristic methods for the flowshop sequencing problem," European Journal of Operations Research, Elsevier, vol. 47, pp. 65–74, 1990.
114. E. Taillard, "Benchmarks for basic scheduling problems," European Journal of Operations Research, Elsevier, vol. 64, pp. 278–285, 1993.
115. D. Todd and P. Sen, "Multiple criteria scheduling using genetic algorithms in a shipyard environment," in Proceedings of the 9th International Conference on Computer Applications in Shipbuilding, K. Johannson and T. Koyama (Eds.), ISBN 4930966027, 1997, pp. 259–274.
116. R. Vaessens, E. Aarts, and J. Lenstra, "Job shop scheduling by local search," INFORMS Journal of Computing, vol. 8, pp. 302–317, 1996.
117. M. Vásquez and L. Whitley, "A comparison of genetic algorithms for the dynamic job shop scheduling problem," in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), D. Whitley et al. (Eds.), Morgan Kaufmann, 2000, Las Vegas, Nevada, USA, July 8–12, 2000, pp. 1011–1018.
118. M. Vásquez and L. Whitley, "A comparison of genetic algorithms for the static job shop scheduling problem," in PPSN VI: Proceedings of the Parallel Problem Solving from Nature Conference, M. Schoenhauer (Ed.), Springer: London, 2000, pp. 303–312.
119. J. Watson, L. Barbulescu, A. Howe, and L. Whitley, "Algorithm performance and problem structure for flowshop scheduling," in Proceedings of the Sixteenth National Conference on Artificial Intelligence, J. Hendler et al. (Eds.), AAAI: Menlo Park, CA, 1999, pp. 688–695.
120. D. Whitley, D. E. Goldberg, E. Cantú-Paz, L. Spector, I. C. Parmee, and H.-G. Beyer (Eds.), in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), Morgan Kaufmann, 2000, Las Vegas, Nevada, USA, July 8–12, 2000.
121. D. Whitley, T. Starkweather, and D. Fuquay, "Scheduling problems and travelling salesmen: The genetic edge recombination operator," in Proceedings of the Third International Conference on Genetic Algorithms, J. D. Schaffer (Ed.), Morgan Kaufmann: San Mateo, 1989, pp. 133–140.
122. D. Whitley, T. Starkweather, and D. Shaner, "Traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination," in Handbook of Genetic Algorithms, L. Davis (Ed.), Van Nostrand Reinhold: New York, 1991, pp. 350–372.
123. D. Wolpert and W. Macready, "No free lunch theorems for optimization," IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67–82, 1997.
124. S. Wu, E. Byeon, and R. Storer, "A graph-theoretic decomposition of the job-shop scheduling problem to achieve scheduling robustness," Operations Research, vol. 47, pp. 113–124, 1999.

125. T. Yamada and R. Nakano, “Scheduling by genetic local search with multi-step crossover,” in Parallel Problem Solving from Nature—PPSN IV, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel (Eds.), Lecture Notes in Computer Science 1141, Springer, 1996, pp. 960–969.
126. J. Zhang, L. Zhao, and W. Kwon, “Scheduling and optimization for a class of single-stage hybrid manufacturing systems,” in Proceedings of the IEEE International Conference on Robotics and Automation, IEEE, 2001, pp. 3115–3120.