Genetic Algorithms for Scheduling [1]

Philip Husbands
School of Cognitive and Computing Sciences
University of Sussex
email: philh@cogs.susx.uk

Abstract

This paper provides a survey of the application of genetic algorithms (GAs) to scheduling. Although it focuses on manufacturing scheduling, particularly job-shop problems, it does outline work in other areas such as transport scheduling and network routing. GA research in closely related problems, such as bin packing and the TSP, are also covered. Finally, it is shown how distributed parallel GAs may allow practically beneficial recharacterisations of highly complex general scheduling problems.

1 Introduction

Practical scheduling problems are numerous and varied. However, many of them share two important characteristics — they are very difficult, and good quality solutions bring highly tangible benefits. In general, scheduling problems are NP-hard [37], consequently there are no known algorithms guaranteed to give an optimal solution and run in polynomial time. This has lead to a long line of techniques emanating from the fields of AI and OR that provide approximate solutions to fairly general classes of problems or exact solutions to highly specific and restricted problems. The former are the more common and tend to rely on the use of heuristics, some form of stochastic optimisation technique, or a mixture of both. This paper will explore the role of genetic algorithms (GAs) within this tradition by examining research past and present, and by attempting to draw conclusions about their most effective use. Much of the work referred to is concerned with manufacturing scheduling. However, the techniques developed for that domain can be adapted to other areas, such as the scheduling of communications networks or project planning, in a straightforward way. It hardly needs saying that scheduling, in its numerous guises, is an immensely important practical problem. It comes as no surprise, then, that it is one of the most popular applications for GA research. Of course scheduling is not particularly glamorous or exciting in itself (for GA applications to capture the imagination see e.g. [7]) but it is an area that has brought forth much interesting GA work. A detailed theoretical analysis of the scheduling problem can be found in Garey and Johnson [37]; well know examples of traditional approaches are those described in Balas [1], Lenstra [33], Carlier and Pinson [3]; general coverage of the subject can be found in Muth and Thompson [38] and French [17]; less traditional AI approaches are described in Ow and Smith [40],and Sycara et al. [46]. Other specific techniques, especially with regard to their relationship with genetic algorithms, will be introduced later in the text.

Because of the complexity of this class of problems, a number of simplifying assumptions have always been used in practical applications. These assumptions are now implicit in what have become the standard problem formulations. Later on in this paper it will be shown that in some circumstances the traditional formulations are far more restrictive than necessary, and further that GA-based techniques may provide a less restrained route forward. However, for the moment we will give the standard formulation of the Job-Shop Scheduling Problem, the most general and most difficult of all traditional scheduling problems, and use it as a formal reference throughout the paper.

1.1 The job-shop scheduling problem

Nearly all practical scheduling problems can be described in terms of the job-shop scheduling problem. Usually as restricted versions of this classic combinatorial optimisation problem. The standard problem definition will be taken to be the following. Consider a manufacturing environment in which n jobs, or items, are to be processed by m machines. Each job will have a set of constraints on the order in which machines can be used and a given processing time on each machine. The jobs may well be of different durations and involve different subsets of the m machines. The job-shop scheduling problem is to find the sequence of jobs on each machine in order to minimise a given objective function. The latter will be a function of such things as total elapsed time, weighted mean completion time and weighted mean lateness under the given due dates for each job. See French [17] or Christophedes [5] for further details of typical objective functions.

More formally, we are given a set $\mathcal{J}$ of n jobs, a set $\mathcal{M}$ of m machines, and a set $\mathcal{O}$ of K operations. For each operation $p \in \mathcal{O}$ there is one job $j_p \in \mathcal{J}$ to which it belongs, and one machine $m_p \in \mathcal{M}$ on which it must be processed for a time $t_p \in \mathbf{N}$. There is also a binary temporal ordering relation $\rightarrow$ on $\mathcal{O}$ that decomposes the set into partial ordering networks corresponding to the jobs. That is, if $x \rightarrow y$, then $j_x = j_y$ and there is no z, distinct from x and y, such that $x \rightarrow z$ or $z \rightarrow y$. Using the minimise makespan objective function, i.e. minimising the elapsed time needed to finish processing all jobs, the problem is to find a start time s_p for each operation $p \in \mathcal{O}$ such that:

$$\max_{p \in \mathcal{O}} (s_p + t_p)$$

is minimised subject to

$$t_p \geq 0, \forall p \in \mathcal{O}$$

$$s_x - s_y \geq t_y, \quad \text{if } y \rightarrow x, \quad x, y \in \mathcal{O}$$

$$(s_i - s_j \geq t_j) \bigvee (s_j - s_i \geq t_i), \quad \text{if } m_i = m_j, \quad i, j \in \mathcal{O}$$

[1] From: AISB Quarterly, No. 89

Job priorities can be handled by instead minimising a weighted sum of processing times.
In fact this is the definition of the *deterministic* job-shop scheduling problem, where all processing times are known exactly and there are no restrictions on when jobs may start. In reality many scheduling problems are not so well defined. The environment may be highly dynamic with new jobs of varying priorities coming in at unpredictable intervals, machines breaking down, or job priorities changing. Processing times and the like may be inherently uncertain. The importance of the dynamic and stochastic facets of the problem will vary from application to application, but in many their full consideration is crucial. A great deal of research in scheduling is limited to the deterministic case; the dynamic stochastic problem proving very difficult to handle. This issue will be returned to later: it is an area where GAs may have something to offer.

1.2 Scope of paper

The major part of this paper, contained in section 2, will review applications of GAs to scheduling and closely related problems. As well as manufacturing examples, this will include a discussion of various approaches to pure sequencing problems as exemplified by the TSP, and some work relating to network routing applications, which have a strong bearing on computer and communications process scheduling. Section 3 draws conclusions about the applicability of GAs to this class of problems.
A basic knowledge of genetic algorithms is assumed throughout. Accessible introductions can be found in the books by Davis [10] and by Goldberg [19].

2 Survey of GA approaches to scheduling and sequencing problems

2.1 Early work

One of the earliest published works on the application of GAs to scheduling is that by Davis [9]. While his paper merely outlines a basic scheme applied to a highly simplified toy problem in flow-shop scheduling (all jobs involve the same processing operations applied in the same order), it contains some interesting and worthwhile material. Davis points out that many real-life scheduling problems involve layers of ill defined constraints that are very difficult, if not impossible, to represent within the formal frameworks demanded by OR techniques. Alternative knowledge-based approaches developed by Fox, Smith and colleagues [35] were able to handle many types of constraints but worked in a deterministic way which often led to highly sub-optimal solutions. Davis conjectured that a GA might be designed that could handle the constraints but by virtue of its stochastic nature would avoid poor local minima.
Very often the key to GA success with practical problems lies in the development of a suitable combination of genotype encoding and genetic operators. Clearly a schedule genotype could simply be a list specifying the order and duration of the operations to be performed by each machine. But simple crossover applied to such strings would nearly always result in illegal offspring with some operations missing, others represented twice, and the flow-shop orderings on the jobs violated. Davis's solution was to use a less literal genotype that was amenable to crossover but required a decoding phase to turn it into a legal solution to the problem. The genotype was a list of preference lists, one for each machine. A preference list consisted of an integer that specified the time the machine should start processing, followed by a permutation of the jobs available and the elements "wait" and "idle". The decoding routine was a simulation of the flow shop's operations. Whenever it had to be decided which operation a machine should perform next, the first available job from its preference list was chosen. The element "idle" forced the machine to remain idle, giving preference to other machines. The element "wait" prevented the machine from processing further along its preference list. The genetic operators employed were as follows: a crossover that exchanged preference lists for selected machines; a scramble operation that randomly reordered members of a preference list; and run-idle, a heuristic operator that inserted "idle" as the second element of the preference list of a machine that had to wait more than an hour for jobs to become available. Each operator was applied probabilistically. The evaluation function used summed the costs of running the flow-shop for five hours with the schedule represented by an individual. Penalty costs were added if jobs were not completed during this interval.
As Davis acknowledges, his model would have to be vastly extended to be used on a realistic problem. He envisaged a methodology based on the close study of deterministic scheduling heuristics to develop more sophisticated encodings and operators. However, his use of domain specific operators and symbolic genotypes, and his implicit call to build GA solutions that make use of existing methods and are able to improve on their performance by providing extra robustness, are important practical contributions that are all to often ignored.
During the early 1980s Fourman experimented with GA methods for tackling layout problems in VLSI [15], problems very closely related to scheduling — indeed they can easily be characterised within an identical mathematical framework. He addresses a symbolic layout problem in which rectangular blocks of fixed sizes, connected by fixed width lines, and subject to various topological and geometric constraints, had to be arranged so as to minimise the total area while violating as few design rules as possible. His genotypes were lists of symbols representing geometric and topological constraints on the positioning of the blocks and connecting lines which were used to determine a layout by feeding them through a deterministic procedure. Standard

crossover, inversion and mutation were readily adapted to operate on these lists. The results obtained were promising but not outstanding. Fourman made some interesting suggestions about using a refined version of the GA systems in which the designer could intervene in the selection process by providing 'hints' based on hard to formalise design knowledge. This sort of approach might lead to the semi-automation of complex design tasks not amenable to other optimisation techniques, because of these unformalizable aspects.

At about the same time Smith and Davis devised a hybrid algorithm, based on a GA, for bin packing [43], that is packing a set of regularly shaped boxes into a fixed space according to some packing density criteria. The work was similar in spirit to Davis's scheduling research described earlier. Again the problem is closely related to scheduling — it is essentially identical to sequencing jobs on a single machine. The genotypes used were simple integer lists determining the order in which the boxes were presented to a deterministic algorithm that went on to do the packing. By combining the GA with the bin-packing algorithm in this way, rather than attempting to explicitly represent the whole layout on the genotype, they produced high quality solutions two orders of magnitude faster than with dynamic programming methods.

As mentioned earlier, in a domain as complex as scheduling there is almost always a need to use heuristics at some level or other, irrespective of which solution technique is being used. However, it is extremely difficult to devise robust and powerful heuristics. Hilliard et al. [22] did some work in attempting to use GA-based classifier machine learning system to learn general scheduling heuristics which might then greatly increase the power of appropriate deterministic search methods. A classifier system employs a set of syntactically simple production rules in conjunction with a reinforcement credit assignment algorithm (to update the strength or worth of rules) and a GA to discover new rules. Full details can be found in Holland and Reitman's seminal paper [24] or Goldberg's book [19]. Hilliard's system was able to discover the classic "sort the jobs by increasing duration" heuristic for the simple one operation per job, no ordering constraints, single machine job-shop scheduling problem. It was not entirely successful on more complex problems but showed promise and points the way to an interesting line of research.

2.2 General sequencing applications

Before going on to describe later developments, it is worthwhile summarizing results in the more general application of GAs to sequencing problems, mainly the TSP, a pure sequencing problem which has a close affinity with scheduling problems, particularly flow-shop scheduling. The task is to find the shortest route through a set of cities, visiting each once only and returning to the starting point. An obvious genotype is a permutation of a list of integers representing the cities. Using this representation, simple crossover would produce illegal tours most of the time, with some cities from the parents represented twice and some not at all. Early efforts by Goldberg [18] and Grefenstette [30] overcame this problem by correcting the offspring tours so that the duplicate cities were replaced by the omitted cities or otherwise eliminated. Reasonable results for small problems were found like this. Among others, Suh [45] later made improvements by incorporating heuristics.

Whitley et al. produced better results by developing a representation and recombination operator that manipulated edges (links between cities) rather than the cities themselves [48]. Their edge recombination operator uses an 'edge map' to construct an offspring that inherits as much information as possible from the parent structures. This map stores all the connections from the two parents that lead into and out of a city. An offspring is started by choosing at random one of the two initial cities from its parents. A tour is then built up by adding a city at a time while favouring those cities with the fewest unused edges (to avoid a city becoming isolated). Candidate 'next' cities will be taken from those connected to the 'current' city in either of the parents (this is the information the edge map holds). This is a good example of using a problem-appropriate representation and recombination operator within the logical framework of the GA. Whitley and his co-workers went on to use this work to develop a prototype production line scheduling system for a Hewlett-Packard board assembly facility [49]. They produced two models: a FIFO system in which the sequence of jobs for the first machine was optimised and then remained the same down the line; and a hybrid system in which the initial sequence of jobs was again optimised, but this time greedy heuristics were used to reorder jobs between subsequent machines. The FIFO and hybrid systems produced very similar results, although the FIFO system was computationally more efficient and converged in fewer generations. However, the published results were for highly structured problems for which better solutions could be constructed by hand. Nevertheless, the results looked promising and comparisons with other algorithms on a range of problems would be very instructive.

Fox and McMahon recently published an insightful paper on genetic encodings and operators for sequencing problems [16]. They introduced a bit string representation for sequences based on a boolean matrix representation of ordering relationships. This scheme can be used for partial orderings as well as the total ordering required for the TSP. A sequence of N elements is represented by a $N \times N$ matrix where each row and each column is uniquely identified with one of the elements. Matrix element $[X, Y]$ contains a 1 iff. symbol X occurs before symbol Y in the sequence. In the n-city TSP case, any matrix representing a sequence must obey three constraints. It must contain exactly $n(n-1)/2$ ones (each city represented exactly once); if $[i, j]$ is one and $[j, k]$ is one then $[i, k]$ must also be one (transitive nature of ordering relation); it must not contain any cycles: $[i, i] = 0\ \forall i$. They introduced two new recombination operators that worked on this representation. Their *intersection* operator was designed to pass on the common characteristics of two parents to the child. The child was formed as follows: first, create the matrix which is the logical AND of the two parent matrices, this will contain all common successor/predecessor relationships; second, add to this a subset of all the ones that are unique to one of the parents; finally convert this underconstrained matrix to a legal sequence by an analysis of the row and column sums. All of these operations are made easy by the matrix representation, and results in an offspring that strongly favours both parents. Their *union* operator is close to traditional crossover but avoids breaking the three basic constraints. It involves four steps: first, partition the set of symbols into two distinct set, S1 and S2; second, construct the matrix containing the bits from the first parent pertaining to S1, and the matrix containing the bits from the second parent pertaining to S2; third perform the logical OR of these two matrices; finally convert this underconstrained matrix to represent a sequence as with the intersection operator. This results in a legal sequence with unique attributes from both parents.

Fox and McMahon compared these operators with Whitley's edge recombination and Goldberg's PMX operator (both mentioned above) among others. They were tested on a 30 city TSP. The operators performed very similarly in terms of quality of solution found. However, the union and intersection operators produced very good solutions in far fewer generations than the other operators although they were significantly more computationally expensive. Although their analysis of encodings for sequencing is certainly instructive, this kind of direct application of GAs remains unproven. Algorithms exist that are capable of finding very good solutions to TSP problems of several thousand cities. In practical terms, algorithms are only useful if they can find solutions of better quality, or more quickly, or more reliably, than with other techniques, or are able to tackle problems previously unmanageable.

Gorges-Schleuter has produced good results for large TSP problems by incorporating local hill-climbing heuristics into a parallel GA [20]. The advantages of parallel GAs will be briefly discussed later.

2.3 Later developments in scheduling with sequential GAs

Cleveland and Smith investigated the use of GAs in scheduling a multi-stage flow line with non-standard characteristics [6]. Interestingly, as well as being one of the pioneers of GA-based machine learning techniques [44], Smith has had much involvement in some influential AI work on heuristic-based scheduling [40, 35]. They investigated a variety of problem formulations and recombination operators, drawing on some of the earlier TSP and scheduling work described above. The problem they addressed is combinatorially complex so previous progress had only been made through the use of heuristics. Their intention was to examine GAs as an alternative approach. They studied three basic models: a pure sequencing version, which assumed that all jobs were available for release at the start of the scheduling horizon, and that work-in-progress cost are negligible; a model that included consideration of actual release times; and a model that also included work-in-progress costs.

In the first version of the problem a comparative analysis of the use of various recombination operators indicated that a number were able to find very good solutions within about 100 generations. Significantly, they found that the GA was able to handle a non-deterministic version of the problem without any apparent loss in performance. The various GAs tested did not perform particularly well on the second model. However, when the more realistic objective function of the third model was used the pure sequencing GAs performed badly whereas GAs employing "schedule-based" representations (in this case Davis's preference lists and a simple bit string representation of the release times of the various jobs) found significantly better solutions.

These results for a complex realistic problem are promising and suggestive. However, further studies on the use of domain knowledge and performance in non-deterministic cases are needed. Careful comparisons with other methods are also required.

The late eighties saw an explosion in the number of GA researchers. Consequently there has been a fair volume of very recent work on scheduling. Some of this will be outlined now, before going on to more advanced techniques. Gabbert et al. successfully applied a GA to transport (train) scheduling and routing [13]. Unlike other approaches, they were able to use modifiable complex cost models, avoiding most of the standard simplifications. They are confident of being able to scale-up their prototype systems. This seems to be a good example of exploiting the strengths of GAs to handle those aspects of a problem that make it unamenable to more traditional techniques.

Wren and Wren have done some very interesting preliminary work on applying GAs to the hard practical problem of bus driver scheduling [50]. Using a straightforward genetic representation of the problem, but with an involved and insightful recombination operator, they were able to find solutions as good as those produced by the best OR techniques. The work stemmed from a desire to find better solutions to the problem, not from a wish to study GAs. There may be a lesson in that. Their paper points out that GAs have largely being ignored by the OR community, and yet here is a significant result from one of the leading researchers on this particular problem.

Another significant and recent result, also coming from the OR community, is that of Dorndorf and Pesch [11]. They use a GA to find optimal sequences of local decisions rules to be used with OR search algorithms. For a range of static deterministic job-shop scheduling problems their hybrid algorithm was able to find shorter makespans (total elapsed time) quicker than Adams, Balas and Zawack's shifting bottleneck procedure [29] and Laarhoven, Aarts and Lenstra's simulated annealing approach [41]. These two techniques were generally regarded as the best available. Dorndorf and Pesch's work is related to earlier research, involving Pesch, on incorporating powerful local search into a GA for the TSP [12].

Mansour and Fox developed a hybrid GA, making use of local hill-climbing and problem specific knowledge, for task allocation in multicomputers [36]. They found significantly better solutions than with a range of other techniques, although the GA was computationally more expensive.

Nakano [39] tackled job-shop scheduling with a genetic encoding similar to that employed by Fox and McMahon (described above) and closely related to Husbands' arbitrator strings [26]. He used simple crossover with a fairly involved genetic repair mechanism to ensure legal offspring. On a set of classic benchmark problems, including the infamous 10×10 and 20×5 problems [38] he was able to find solutions which compared very favourably with state-of-the art branch and bound techniques. However, no comparative results on computational resources needed were given, and the genetic repair method, as described, appears computationally expensive.

Syswerda describes a GA-based system for scheduling the use of laboratory equipment [47]. He employed a GA to find an initial sequence of tasks to feed to a fairly sophisticated deterministic schedule builder such that near optimal schedules result. His genotype was simply a list representing a task permutation. He used various mutation operators: select two tasks at random, place the second before the first; select two tasks at random, interchange their positions; scramble a randomly chosen sub-list of the genotype. He experimented with order and position based crossovers as well as Whitley's edge recombination operator. This hybrid approach, where the GA works in tandem with a deterministic search method, produced good results fast enough that it could take in the dynamic aspects of the problem and allowed rescheduling.

Reeves has done some preliminary experiments on applying GAs to stochastic flow-shop problems [42]. Over a range of different problem instances his algorithm consistently outperformed two other techniques from the OR literature.

Ling was able to find good solutions to a large college timetabling problem by first using a heuristic-based algorithm to build a reasonable timetable, but with some constraints violated, and then applying a GA to convert this into a solution with no constraints broken [34].

Bruns has recently had some success with a slight twist on the hybrid GA theme [2]. His genotypes are direct symbolic representations of a production schedule, but his genetic operators are highly specialised, incorporating a good deal of domain knowledge.

Juliff [31] has developed an interesting multi-chromosome GA for multi-dimension scheduling problems (she uses pallet loading as an example). Key dimensions of the solution (e.g. in her example: layer order, pallet type, and pallet order) are represented on different chromosomes. She has found this method to be superior to a single chromosome representation.

Fang et al. [14] found that a variant of Grefenstette's ordinal representation for the TSP [30] worked very efficiently for standard job-shop problems.

2.4 Parallel GAs

From the very earliest days of its development the GA's potential for parallelisation, with all its attendant benefits of efficiency, has been noted. The availability of hardware has recently allowed significant progress in this direction. The standard sequential GA uses global population statistics to control selection, so the processing bottleneck is evaluation. The earliest parallel models simply parallelised this phase of the sequential algorithm, see, for instance, the paper by Grefenstette [21]. Recently more sophisticated parallel GAs have started to appear in which population can be viewed as being spread out geographically, usually over a 2D toroidal grid. All interactions, e.g. selection and mating, are local, being confined to small (possibly overlapping) neighbourhoods. By doing away with global calculations, this allows the development of fine-grained highly parallel asynchronous algorithms. There is mounting evidence to suggest that such systems are more robust and faster (in terms of solutions evaluated) than other implementations, e.g. see the articles by Collins & Jefferson [8] and Husbands [25]. Highly parallel models can also result in powerful new ways of approaching optimisation problems at the conceptual level, as the following two sections illustrate.

2.5 Parasites and Sorting Networks

Danny Hillis was the first to significantly extend the parallel GA paradigm by showing how to develop a more powerful optimisation system by making use of coevolution [23]. He has found that locally controlled selection is more robust than the simple global variety. Specifically, in his experiments individuals evolve on a 2D toroidal grid with the x and y displacements of an individual from potential mates being a binomial approximation of a Gaussian distribution. After a pair mate, the two offspring they produce replace them, in the same locations, so the genetic material remains spatially local.

An interesting complex optimisation problem that he has tackled using GAs, is the problem of finding minimal sorting networks for a given number of elements. A sorting network represents a sorting algorithm in which comparisons and exchanges take place in some predetermined order, see Knuth [32] for further details. Finding good networks is of significant practical interest, bearing on the development of optimal sorting algorithms, switching circuits and, particularly pertinent to this paper, network routing algorithms. A sorting network is represented in Figure 1. The horizontal lines correspond to the elements to be sorted. The unsorted input is on the left and the sorted output is on the right. In between, comparison-exchanges of elements are indicated by arrows pointing from one element to another. A comparison-exchange of the ith and jth elements is indicated by an arrow from the ith to the jth line. Elements are exchanged if the element at the head of the arrow is strictly less than the element at the tail. The network shown in the figure is random.

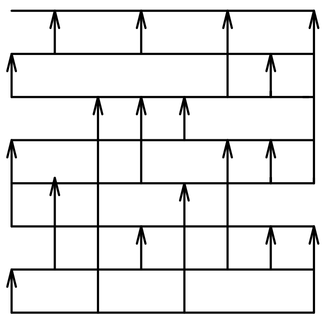

Figure 1: A sorting network.

The genotype of each individual consisted of a pair of bit string chromosomes Each chromosome can be thought of as sixty eight-bit genes. Each gene consisted of two four-bit numbers representing elements to be compared and possibly exchanged. The phenotype (sorting network) is generated by traversing the chromosomes in a fixed order. If a pair of chromosomes have the same gene (comparison-exchange elements) at a particular site, then only that comparison-exchange pair is generated in the sorting network. If the genes are different, then both pairs are generated. In this way the network can contain between sixty and one hundred and twenty comparison-exchanges. The phenotypes were scored according to how well they sorted. The measure used was the percentage of correct sorts performed on a sample of test cases. The best results were produced when the test cases were *coevolved* along with the sorters, rather than being randomly generated. This is analogous to the biological evolution of a host-parasite system. In this extended model there are two independent gene pools, each evolving according to local selection and mating. One population, the hosts, represents sorting networks, the other, the parasites, represents test cases. Interaction of the populations is via their fitness functions. The sorting networks are scored according to the test cases provided by the parasites in their immediate vicinity. The parasites are scored according to the number of tests the network fails on. The best result found was a network requiring sixty one exchanges, only one more than the best known. Significantly, by using an advanced distributed approach Hillis was able to tackle a highly complex problem to which previous solutions had been hand-crafted over many years.

2.6 Symbiosis and Emergent Scheduling

The underlying structure of many combinatorial optimisation problems of practical interest is highly parallel. However, traditional approaches to these problems tend to use mathematical characterisations that obscure this. By contrast, the use of biologically inspired models casts fresh light on a problem and may lead to a more general characterisation which clearly indicates how to exploit parallelism and gain better solutions. This section describes a model based on simulated coevolution that has been applied to a highly generalised version of the job-shop scheduling problem, far more general than any of the models presented so far. Whereas Hillis's model is analogous to a host-parasite ecology, this model is closer to a symbiotic ecology. That is, a number of separate species interacting in ways that are to their mutual advantage. Full details can be found in [28, 26, 27].

In the standard model of manufacturing planning, process planning directly proceeds scheduling. A process plan is a detailed set of instructions on how to manufacture each part (process each job). This is when decisions are made about the appropriate machines for each operation and any constraints on the order in which operations can be performed (see Chang & Wysk for further details [4]). Very often completed process plans are presented as the raw data for the scheduler. Scheduling is essentially seen as the task of finding an optimal way of interleaving a number of fixed, or maybe slightly flexible, plans which are to be executed concurrently and which must share resources. However, in many manufacturing environments there are a vast number of legal plans for each component. These vary in the orderings between operations, the machines used, the tools used on any given machine and the orientation of the work-piece on any given machine. They will also vary enormously in their costs. Instead of just generating a reasonable plan to send off to the scheduler, it is desirable to generate a near optimal one. Clearly this cannot be done in isolation from the scheduling: a number of separately optimal plans for different components might well interact to cause serious bottle-necks. Because of the complexity of the overall optimisation problem, that is simultaneously optimising the individual plans and the schedule, and for the reasons outlined in the introduction, up until now very little work has been done on it. However, recasting the problem to fit an 'ecosystem' model of coevolving organisms has provided a promising new direction.

Husbands' model involves a number of different populations coevolving on a 2D grid with local selection in force. The genotype of each specie represents a feasible process plan for a particular component to be manufactured in the machine shop. Separate populations evolve under the pressure of selection to find near-optimal process plans for each of the components. However, their fitness functions take into account the use of shared resources in their common world (a model of the machine shop). This means that without the need for an explicit scheduling stage, a low cost schedule will emerge at the same time as the plans are being optimised. The data provided by a *plan space generator*, whose operation is described in [26], is used to randomly generate populations of structures representing possible plans, one population for each component to be manufactured. An important part of this model is a population of Arbitrators, again initially randomly generated. The Arbitrators' job is to resolve conflicts between members of the other populations; their fitness depends on how well they achieve this. Conflicts arise when plans from different populations demand the same resource (e.g. machine) at the same time. There is a single representative of each population in each cell on the grid. Each genotype is costed according to its use of resources and its interactions with the other plans in its cell. Combined with local selection, this allows for a coherent coevolution and has resulted in good results for a number of large test problems, including ones involving a noisy dynamic workshop [27].

3 Conclusions

There is a growing tradition in applying GAs to a wide variety of scheduling problems. Much of the early work was concerned with using the GA to tackle the whole problem (in practice often the TSP). While this produced some useful insights into genetic coding schemes and operators, it was of little immediate practical value. Very often the main concern of the researchers involved was to gain a deeper understanding of the workings of GAs rather than to solve real problem. More recently GAs have been used in tandem with other methods to tackle complex realistic problem. Results from this area of investigation are extremely encouraging. Highly parallel distributed GAs have provided completely new ways of looking at generalised scheduling problems and show promise in handling dynamic and stochastic problems. In conclusion, if a proven well understood deterministic technique exists for the problem you wish to tackle, then you should use it. However, if your problem is uncertain and contains aspects difficult to formalise and is not served well by traditional methods, then a GA used with subtlety, insight, and quite possibly in conjunction with other techniques, may provide a good route forward.

References

[1] E. Balas. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. *Operations Research*, 17:941–957, 1969.

[2] R. Bruns. Direct chromosome representation and advanced genetic operators for production scheduling. In S. Forrest, editor, *Proc. 5th Int. Conf. on GAs*, pages 352–359. Morgan Kaufmann, 1993.

[3] J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35(2):164–176, 1989.

[4] T. Chang and R. Wysk. *An introduction to process planning.* Prentice-Hall, 1985.

[5] N. Christophedes. *Combinatorial Optimisation.* Wiley, 1979.

[6] G. Clevland and S. Smith. Using genetic algorithms to schedule flow shop releases. In J.D. Schaffer, editor, *Proc. 3rd Int. Conf. on GAs*, pages 160–169. Morgan Kaufmann, 1989.

[7] D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors. *From animals to animats 3: Proc. 3rd Int. Conf. on simulation of adaptive behavior.* MIT Press, 1994.

[8] R. Collins and D. Jefferson. Selection in massively parallel genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, ICGA-91*, pages 249–256. Morgan Kaufmann, 1991.

[9] L. Davis. Job-shop scheduling with genetic algorithms. In J. Grefenstette, editor, *Proc. Int. Conf. on GAs*, pages 136–140. Lawrence Erlbaum, 1985.

[10] L. Davis. *The Handbook of Genetic Algorithms.* Van Nostrand Reinhold, 1990.

[11] U. Dorndorf and E. Pesch. Evolution based learning in a job-shop scheduling environment. Technical Report RM-92-019, Faculty of Economics, Limburg University, 1992.

[12] N. Ulder et al. Genetic local search algorithms for the travelling salseman problem. In H. Schwefel and R. Manner, editors, *Parallel problem solving from nature*, pages 105–109. Springer Verlag, LNCS-496, 1991.

[13] P. Gabbert et al. A system for learning routes and schedules with genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, ICGA-91*, pages 430–436. Morgan Kaufmann, 1991.

[14] H. Fang, P. Ross, and D. Corne. A promising genetic algorithm approach to job-shop scheduling, re-scheduling, and open-shop scheduling problems. In S. Forrest, editor, *Proc. 5th Int. Conf. on GAs*, pages 375–382. Morgan Kaufmann, 1993.

[15] M. Fourman. Compaction of symbolic layout using genetic algorithms. In J. Grefenstette, editor, *Proc. Int. Conf. on GAs*, pages 136–141. Lawrence Erlbaum, 1985.

[16] B. Fox and M. McMahon. Genetic operators for sequencing problems. In G. Rawlins, editor, *Foundations of genetic algorithms.* Morgan Kaufmann, 1991.

[17] S. French. *Sequencing and scheduling: an introduction to the mathematics of the job-shop.* Ellis Horwood, 1982.

[18] D. Goldberg and R. Lingle. Alleles, loci and the travelling salesman problem. In J. J. Grefenstette, editor, *Proc. Int. Conf. on GAs*, Hillsdale, N.J., 1985. Lawrence Erlbaum Associates.

[19] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, Massachusetts, USA, 1989.

[20] M. Gorges-Schleuter. Asparagos: An asynchronous parallel genetic algorithm. In J.D. Schaffer, editor, *Proceedings of the Third Intl. Conf. on Genetic Algorithms, ICGA-89*, pages 422–427. Morgan Kaufmann, 1989.

[21] J. Grefenstette. Parallel adaptive algorithms for function optimisation. Technical Report CS-81-19, Compt. Sci., Vanderbilt University, 1981.

[22] M. Hilliard. A classifier based system for discovering scheduling heuristics. In J. Grefenstette, editor, *Proceedings of the second international conference on GAs*, pages 231–235. Lawrence Erlbaum, 1987.

[23] W.D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990.

[24] J. Holland and J. Reitman. Cognitive systems based on adaptive algorithms. In P. Waterman and F. Hayes-Roth, editors, *Pattern directed inference systems*. Academic Press, 1978.

[25] P. Husbands. Genetic algorithms in optimisation and adaptation. In L. Kronsjo and D. Shumsheruddin, editors, *Advances in Parallel Algorithms*, pages 227–277. Blackwell Scientific Publishing, Oxford, 1992.

[26] P. Husbands. An ecosystems model for integrated production planning. *Intl. Journal of Computer Integrated Manufacturing*, 6(1&2):74–86, 1993.

[27] Philip Husbands. Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. In T. Fogarty, editor, *Evolutionary Computing, AISB Workshop Selected Papers*, pages 150–165. Springer-Verlag, Lecture Notes in Computer Science Vol. 865, 1994.

[28] Philip Husbands and Frank Mill. Simulated co-evolution as the mechanism for emergent planning and scheduling. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, ICGA-91*, pages 264–270. Morgan Kaufmann, 1991.

[29] E. Balas J. Adams and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3):391–401, March 1988.

[30] B. Rosmaita J. Grefenstette, R. Gopal and D. Van Gucht. Genetic algorithms for the travelling salesman problem. In J. J. Grefenstette, editor, *Proc. Int. Conf. on GAs*, Hillsdale, N.J., 1985. Lawrence Erlbaum Associates.

[31] K. Juliff. A multi-chromosome genetic algorithm for pallet loading. In S. Forrest, editor, *Proc. 5th Int. Conf. on GAs*, pages 467–473. Morgan Kaufmann, 1993.

[32] D. Knuth. *Sorting and searching. Vol. 3, The art of computer programming*. Addison-Wesley, 1973.

[33] J. Lenstra. Sequencing by enumerative methods. Technical report, Mathenatisch Centrum, Amsterdam, 1976.

[34] S. Ling. Constructing a college timetable by integrating two approaches: logic programming and genetic algorithms. Technical Report MSc. Thesis, School of Cognitive and Computing Sciences, University of Sussex, 1991.

[35] S. Smith M. Fox, B. Allen and G. Strohm. Isis: A constraint-directed reasoning approach to job-shop scheduling. Technical report, Robotics Institute, CMU, 1983.

[36] N. Mansour and G. Fox. A hybrid genetic algorithm for task allocation in multicomputers. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, ICGA-91*, pages 466–473. Morgan Kaufmann, 1991.

[37] M.Garey and D.Johnson. *Computers and intractability: a guide to the theory of NP-Completeness*. W.H.Freeman, 1979.

[38] J. Muth and G. Thompson. *Industrial Scheduling*. Prentice-Hall, 1963.

[39] R. Nakano and T. Yamada. Conventional genetic algorithms for job-shop problems. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, ICGA-91*, pages 474–479. Morgan Kaufmann, 1991.

[40] P. Ow and S. Smith. Viewing scheduling as an opportunistic problem solving process. *Annals of Operations Research*, 12, 1988.

[41] E. Aarts P. Laarhoven and J. Lenstra. Job shop scheduling by simulated annealing. Technical Report OS-R8809, Centre for mathematics and computer science, Amsterdam, 1988.

[42] C. Reeves. A genetic algorithm approach to stochastic flowshop sequencing. In *Colloquium on genetic algorithms for control and systems engineering*, volume 1992-106 of *IEE Digest Series*, pages 13/1–13/4, London, 1992. IEE.

[43] D. Smith. Bin packing with adaptive search. In J. Grefenstette, editor, *Proc. Int. Conf. on GAs*, pages 202–207. Lawrence Erlbaum, 1985.

[44] Stephen F. Smith. *A Learning System based on Genetic Adaptive Algorithms*. PhD thesis, Department of Computer Science, University of Pittsburgh, USA, 1980.

[45] J. Suh and D. Van Gucht. Incorporating heuristic information into genetic search. In J. Grefenstette, editor, *Proc. 2nd Int. Conf. on GAs*, pages 100–108. Lawrence Erlbaum, 1987.

[46] K. Sycara, S. Roth, and M. Fox. Resource allocation in disributed factory scheduling. *IEEE Expert*, pages 29–40, Feb. 1991.

[47] G. Syswerda. Schedule optimisation using genetic algorithms. In L. Davis, editor, *The Handbook of Genetic Algorithms*, pages 332–349. Van Nostram Reinhold, 1990.

[48] D. Whitley and T. Starkweather. Genitor II: A distributed genetic algorithm. *Journal of Experimental and Theoretical AI*, 2:189–214, 1990.

[49] D. Whitley, T. Starkweather, and D. Shaner. The travelling salesman and sequence scheduling: quality solutions using genetic edge recombination. In L. Davis, editor, *The Handbook of Genetic Algorithms*, pages 350–372. Van Nostram Reinhold, 1990.

[50] A. Wren and D. Wren. Genetics, structures and covers – an application to scheduling. Technical Report 90.23, School of Computer Science, University of Leeds, 1990.