Using Genetic Algorithms in Industry – Art or Science?

Hugo Duncan
Alcan Inc.
1188 Sherbrooke St West
Montreal, Quebec, H3A 3G2
hugo.duncan@alcan.com

Gerard Leconte
Alcan Inc.

Peter Utiger
Alcan Inc.

Abstract

We describe scheduling in the metals processing industry and present some of the challenges that we have faced in applying genetic algorithms to provide real world scheduling systems for use in our factories. After examining the technical difficulties in implementing real world genetic engines, we conclude that the use of genetic algorithms in industry is still an art, not a science.

1 SCHEDULING FOR METALS PROCESSING INDUSTRY

Our group first used a genetic algorithm (GA) as part of a scheduling system put into a fabrication plant in North America, seven years ago. Since then we have built on this experience and now have GA driven scheduling systems in a continuous casting plant in Canada, on hot mills in North America and Korea, on many cold mills, and on a paint-line in Italy.

Our experience has led to a pragmatic approach based on heuristics, with the aim of producing schedules that are "good enough".

We begin by giving a broad description of the processes we are scheduling, describe where GA's fit in a real world scheduling system, and then examine the technicalities of implementation.

We have included significant detail, restricted in some places by our need to respect the proprietary nature of the applications.

1.1 THE SCHEDULING PROBLEM

The manufacturing process in our factories is based on the working of a single piece of metal through multiple machines, each of which changes either some physical property of material, such as the thickness, or some metallurgical property, such as the hardness.

The single-machine scheduling problem is thus based on sequencing a set of materials such that the sequence can be produced on the machine, and is optimised in some way. The sequencing problem is often complicated by the need to group items together, as some machines process the metal pieces in batches. The sequence will give rise to delays between the processing of successive pieces of material, in order to carry out some sequence dependent set-up operations.

Usually, the sequence will also affect the quality of the product, particularly when the thermal state of the machine has a time constant of the order of the processing time of several pieces of material.

Material is made to order, and one-off orders are common. An order specifies a particular quantity of a given specification. For a given machine, there may be between 100 and 500 pieces of material processed each day, with anything from one piece per specification to 10 pieces of a given specification. This is a relatively small number of pieces, and necessitates a scheduling horizon of several days to obtain scope for optimisation.

The scheduling system often has an additional role to fulfil, assigning material to specific orders (specifications). In general, a single specification of input material can be used to make several different types of output material and the type to be produced must be chosen. Several orders may be made from one material, or many pieces of material may be required for a single order. Thus each job has an earliest availability determined by the material assigned to it, and a latest start determined by the required delivery date, and this additionally constrains the problem.

This structure is present in the aluminium fabrication business across most of the machines in the production route, e.g. hot mill, caster, slitter, or paint line.

1.2 CHARACTERISTICS

The problem has a multi-level structure, involving the sequencing, grouping and assignment of pieces of material. These problems are not separable, and must be solved together. Within a specific scheduling problem, the group size may vary considerably (e.g. from one to 20 pieces), depending on the product and the order volumes.

The sequence dependent set-ups are frequent (e.g. every ten pieces in an optimised schedule) and are caused by many different process rules. The duration of the set-up is relatively independent of the reason for the set-up, and it is difficult to alter the schedule to remove a delay for one reason, without introducing a similar delay for another reason. This gives rise to a fitness landscape that consists of narrow peaks and wide valleys (at least that is our intuition, see later).

It is relatively easy to optimise one portion of a schedule, so we would expect the fitness landscape to also demonstrate many local optima.

2 REAL WORLD SCHEDULING SYSTEM

A real world scheduling system is one component of an overall system designed to achieve the business objectives of maximising on time delivery, maximising throughput, and minimising work in process. A scheduling system plays a key role in this, but can only succeed in coordination with planning and manufacturing execution systems.

We estimate that the implementation of an optimised schedule builder using a GA represents less than 20% of the effort required to build a real world scheduling system. For our projects we have written an implementation of a GA that can be used in all of our scheduling systems and reduces the effort for new systems below even this level. The remaining time is spread over the items below, which are included here to show the level of effort required in implementing a real world scheduling system, relative to the effort involved in the GA optimiser.

2.1 SCHEDULING RULE FORMALISATION

One of the longest processes in building a scheduling engine is the elicitation of the scheduling rules, and their formalisation in a suitable form in code. These rules determine the permissibility and the quality of a schedule and are often highly specific and numerous.

The compilation and encoding of the rules is a major part of the implementation effort. Combined with the fact that these rules often reflect proprietary practices, this may explain why industry has not produced many benchmark problems for use in GA scheduling research.

2.2 SCHEDULE EDITING

The person responsible for scheduling requires the ability to manually construct a schedule and to edit schedules that have been generated by the optimising engine. This enables the human scheduler to maintain responsibility for the schedules, reacting to minor events on the factory floor, and correcting any constraints that have not been implemented in the system.

2.3 INTERACTION WITH OTHER SYSTEMS

The GA scheduler uses the order book, material inventory and machine schedules and links to the manufacturing systems that provide these must be established for the GA scheduler to operate. This often involves synchronising data from several disparate systems.

2.4 SOLUTION TIME

For a scheduling system to respond to unexpected events on the factory floor, it needs to be able to produce schedules within 1/2 hr on a current PC.

3 GA IMPLEMENTATION ISSUES

In creating a GA for use as the optimisation engine in our scheduling systems, we are faced with a number of design decisions.

3.1 BLACK BOX OR CUSTOM GA

While there has been extensive recent work on developing black box GA's (Pelikan et al. 2001, Bosman and Thierens, 2001), we feel that this work has yet to become advanced enough to handle the type of problem described above. The major issue facing these systems is learning the problem specific structure and we have decided to work with a custom GA, allowing us to add the problem domain knowledge directly into the GA, reducing the requirement for a learning algorithm.

We note that most of the learning schemes are based on building probabilistic models, and have yet to use an adaptive solution representation that would enable direct coding and propagation of identified linkage.

3.2 SCHEDULE REPRESENTATION

Having decided on a custom GA, we have also always decided on a problem specific representation for our genetic optimisers.

Some theoretical considerations have been proposed on why composite data structures might be preferable (Radcliffe 1992). Genetic programming (GP) in practice (Koza 1999) has shown the usefulness of a graph representation for specific problems. The vast body of work on GP shows the advantages of using this representation and associated operators when suited to the problem at hand. Work on grouping algorithms (Faulkener 1998) has shown the advantages of problem specific representation in that domain.

From a practical viewpoint, designing a representation allows close control over the characteristics of the projection from search space to schedule space. In particular we strive for a bijective projection, with one possible coding of every possible schedule, with every possible representation mapping to a valid schedule. Achieving this aim greatly simplifies the writing of operators that change an individual.

The custom representation allows direct inclusion of the problem linkage in the coding. This not only obviates the need to learn the linkage, but also allows the building blocks to be directly propagated by the representation.

The disadvantage of using a custom representation is that we give up all recourse to formal results on operator design, selection pressure, convergence, etc – we effectively give up all current results. While we expect that the general tendencies expressed by the formal results on simple GA's will continue to apply, this interpretation is in the realms of design as an art, and requires considerable experimentation.

3.3 GENETIC OPERATORS

Writing the operators and selecting their parameters is 30% of the work of implementing the GA. The design of the operators spans all representation, schedule and objective spaces and has a determining impact on the performance of the schedule optimiser.

It is well known from schema theory that the operators bias the search space, and that biasing the search space is what makes hard problems tractable.

While crossover tends to be implemented as a modification of the traditional crossover operators, our mutation operators are much more varied, and play a role in both the exploration of the landscape and in the refinement (i.e. local search) of promising solutions.

The mutation operators are often designed based on heuristics used in manual schedule construction, and ensure that the schedule optimiser is capable of generating key features of "good schedules".

At present the design of the mutation operators is based on trial and error. As far as we are aware, there are no tools available to help analyse operator performance, either theoretically or empirically. We would like to be able to answer questions on the effect of the operators on the landscape (are they local search operators or exploration operators), the bias they introduce, how orthogonal they are to each other, and whether the operators allow the exploration of all (or at least the good parts) of the landscape.

We have no way of answering these questions at the moment, which increases considerably the skill required in build a real world application. The development of schema theory (Poli, Langdon, Vose) promises to deliver some theoretical answers for standard encoding and operators, but for the real world we need empirical tools to be able to apply schema analysis to the design of our genetic optimisers.

3.4 SELECTION PRESSURE

Linked to the choice of operators is the wider issue of selection pressure, determined by selection algorithms and operator probabilities (Goldberg and Sastry 2001).

We take qualitative notice of the conclusions in the theory and then resort to trial and error. We utilise many mutation operators, and a design methodology to address the questions of absolute level of mutation and the relative weights of the operators would be of considerable benefit.

3.5 EVALUATION

Writing an evaluation function for decoding the representation is 70% of the work of implementing the GA. The evaluation function applies many of the process rules, and allows the objective functions to work with real world quantities, such as delay times, machine throughput, delivery delays, etc.

3.5.1 Constraints

One approach used in the literature is to use a separate objective for constraints. Due to the limitation above we use the approach of having hard constraints only (i.e. we ensure that all schedules considered are valid).

Soft constraints represent a heuristic on what an optimised schedule is, and are included through penalising one of the main objective functions. This has however leads to large flat valleys in the landscape, due to the number of constraints that can have the same penalty.

3.6 OBJECTIVE FUNCTIONS

Scheduling problems are by definition multi-objective in nature and we have drawn heavily on work in this area (Fonseca & Fleming 1995, Coello 1999), in particular on the MOGA.

The objective functions are chosen to represent quantities that are managed by the schedulers, such as on-time delivery, machine productivity and material inventory.

For a manageable optimisation time, we have found that a maximum of three objectives is a practical limit. This constraint arises due to the increased population size required to get dense enough coverage of the Pareto front. For N orthogonal objectives, the Pareto front is generally a N-1 dimensional surface, and we conjecture that $O(x^N)$ population size is required given a single objective population x calculated from other sizing requirements.

Since the objective functions are real world, we have little influence on the nature of the mapping between schedule and objective space. There is no guarantee that the projection will be one-to-one, and there is no guarantee that the projection is continuous.

This is problematic when designing and tuning operators, as we have no method for visualising the search space. Without such a visualisation it is impossible to develop separate knowledge of the landscape and the action of the operators, leaving us to use what we can infer from the convergence dynamics of the optimiser as a whole.

A visualisation might rely on some concept of distance between solutions for a permutation problem. While

there are some ideas for calculating distance between sequences to be found in DNA sequencing, these ideas have not yet been applied to scheduling GA's.

4 CONCLUSIONS

We show that that the generation of optimised schedules is one part of a scheduling system, which in turn is part of overall planning and manufacturing system designed to meet the business objectives of on-time delivery, productivity and material inventory.

The design decisions taken to build a system to obtain our pragmatic goal of generating adequate schedules within a reasonable time have taken us away from a solution based on a standard GA. Once we leave the standard GA the direct applicability of published work diminishes, and we revert to a heuristic solution of the issues involved.

We would welcome design tools to support the empirical design of our GA's. The ability to visualise the search space for scheduling type problems would aid our understanding of the landscape, enabling us to design better operators. Tools to characterise the performance of operators when applied individually and in interaction with other operators would speed the development process and lead to improved solution quality.

The infrequent use of GA's in industrial scheduling can probably be attributed to the relatively small role the optimisation engine plays in an scheduling system, and the complexity and art of implementation, despite the undeniable benefits that we have seen from these systems.

Acknowledgments

We would like to thank Benoit Hap and Renaud Dumeur at Computers Communications & Visions (C^2V) for introducing us to genetic algorithms and providing initial implementations and impetus for our work.

References

P.A.N.Bosman and D Thierens (2001). Crossing the Road to Efficient IDEAS. In proceedings, *Genetic and Evolutionary Computation Conference*, 219-226.

C.A.C.Coello 1999. An Updated Survey of Evolutionary Multiobjective Optimization Techniques: State of the Art and Future Trends. 1999 Congress on Evolutionary Computation.

E.Faulkener (1998), Genetic Algorithms and Grouping Problems, John Wiley & Sons.

C.M.Fonseca & P.J.Fleming (1995) An Overview of Evolutionary Algorithms in Multiobjective Optimisation. Evolutionary Computation, 3(1), 1-16.

D.E.Goldberg and K Sastry (2001). A Practical Schema Theorem for Genetic Algorithm Design and Tuning. In proceedings, *Genetic and Evolutionary Computation Conference*, 328-335

J.R.Koza (1999). Genetic Programming III; Darwinian invention and problem solving. Morgan Kaufmann Publishers.

Pelikan, M., Sastry, K., Goldberg, D.E. (2001) Evolutionary Algorithms + Graphical Models = Scalable Black-Box Optimization. IlliGAL report 2001029

N.J.Radcliffe (1992). Non-Linear Genetic Representations. Parallel Problem Solving from Nature 2, R. Manner and B. Manderick (Ed.), (Elsevier Science Publishers, Amsterdam), 259-268.