G9.5 Experiments with an ecosystems model for integrated production planning

Philip Husbands, Malcolm McIlhagga and Robert Ives

Abstract

This paper outlines a coevolutionary distributed genetic algorithm for tackling an integrated manufacturing planning and scheduling problem. In this multispecies ecosystems model, the genotype of each species represents a feasible manufacturing (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. Results are presented of the use of this model on a set of industrial problems. It is shown to significantly outperform simulated annealing and a dispatching rule algorithm over a wide range of optimisation criteria.

G9.5.1 Project Overview

Research on job shop scheduling (JSS), as the most general of the classical scheduling problems, has generated a great deal of literature (Muth and Thomson 1963, Balas 1969, Garey *et al* 1976, Graves 1981, Ow and Smith 1988, Carlier and Pinson 1989). All of this work has used a particular definition of the scheduling problem or very close variants of it. This article describes a case study where a multispecies coevolutionary genetic algorithm is used to tackle a less restricted highly generalised version of JSS. It is shown how the technique provides an integrated production planning system, treating process planning and scheduling as inextricably interwoven parts of the same problem.

The traditional view of JSS is shown in figure G9.5.1. A number of *fixed* manufacturing plans, one for each component to be manufactured, are interleaved by a scheduler so as to minimise some criteria such as the total length of the schedule. 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:

$$\max_{p \in \mathcal{O}}(s_p + t_p) \tag{G9.5.1}$$

is minimised subject to:

$$t_p \geq 0, \forall p \in \mathcal{O} \tag{G9.5.2}$$

$$s_x - s_y \geq t_y, \quad \text{if } y \rightarrow x, \quad x, y \in \mathcal{O} \tag{G9.5.3}$$

$$(s_i - s_j \geq t_j) \bigvee (s_j - s_i \geq t_i), \quad \text{if } m_i = m_j, \quad i,j \in \mathcal{O} \qquad \text{(G9.5.4)}$$

However, a problem that would often be more useful to solve is that illustrated in figure G9.5.2. Here the intention is to optimise the individual manufacturing plans *in parallel*, taking into account the numerous interactions between them resulting from the shared use of resources. This is the optimization task that henceforth will be termed the integrated planning and scheduling problem and is the focus of this case study. An ecosystems model has been developed to tackle various practical instances of this problem, one of which is presented here.

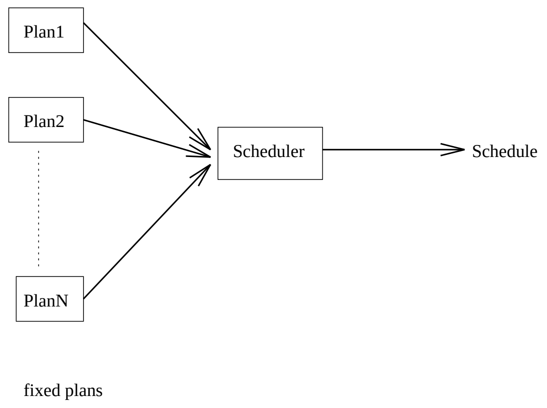

Figure G9.5.1. Traditional approach to job shop scheduling.

The idea behind the ecosystems model is as follows. The genotype of each species represents a feasible manufacturing (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 system is illustrated in figure G9.5.3. The role of the Arbitrators, which coevolve along with the other species, is to resolve resource conflicts betwen manufacturing plans for different components.

This project is one of the strands of ongoing research in the Evolutionary and Adaptive Systems Group, School of Cognitive and Computing Sciences, University of Sussex. It has been carried out in collaboration with Edinburgh University, Logica, and Rolls Royce.

Description of the Problem

The integrated planning and scheduling problems considered in this case study are typical industrial problems. They are generated from data collected from David Brown Vehicle Transmissions Ltd. They model the manufacture of medium complexity prismatic parts, by metal removal processes. They are based on the work of Palmer (1994).

The statistics shown in section G9.5.4 are all mean figures taken from 100 sample problems. A problem consists of a number of jobs (1-14 jobs for each problem) each of which requires a plan and all of which must be scheduled for a specific shop-floor. A job is assumed to be one or more identical parts which (usually) remain together as they move through the shop floor. Here each part could have 1-14 processes. A part consists of a blank (the raw material that it is machined from) and a number of features which define its appearance, these can be thought of as describing volumetric removals of material from the blank. A process plan for a given part may be either fixed or flexible, either way the process plan describes the processes that must be carried out (including possible ordering or sequencing constraints) for a specific set of features to appear on the work-piece. However, the process plan does not define the exact way in which that feature is to be machined.

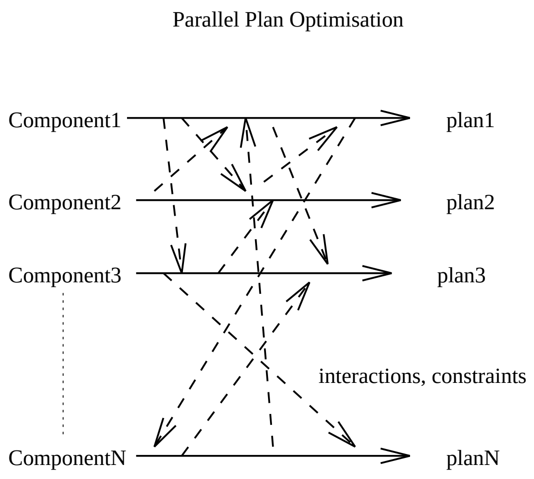

Figure G9.5.2. Parallel plan optimisation leading to emergent scheduling.

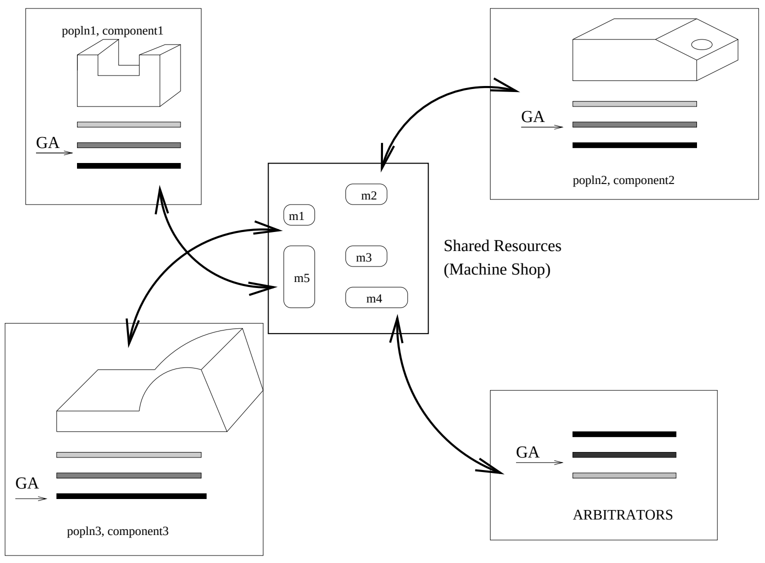

Figure G9.5.3. The ecosystems model.

The GA searches for near-optimal combinations of processes, machines, tools and setups (work-piece orientations) for each feature, taking into account interactions with other features and the overall constraints of the problem. In this case the shop-floor does not alter between problems. The shop-floor consists of 25 machines which vary in the number and diversity of processes that they can carry out. Each process plan is generated from the defined object (including some description of its features and certain possible machining order constraints) and the possible processes that can generate those features on the work- piece; in this case there are one or two applicable processes per feature. For full details see (Palmer 1994, McIlhagga *et al* 1995).

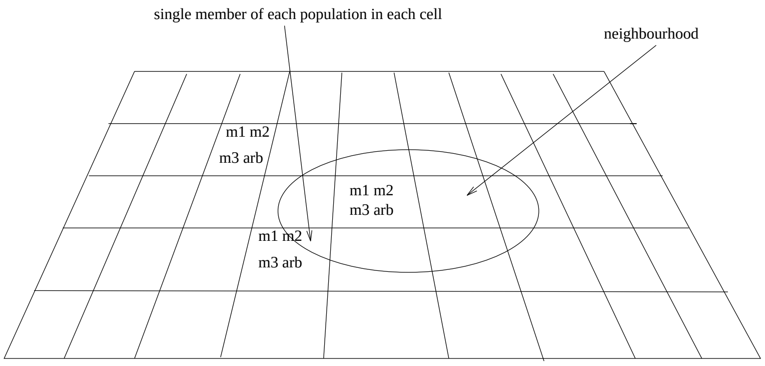

Figure G9.5.4. Distributed interacting populations.

G9.5.2 Design Process

Motivation

It is well known that the standard JSS problem is NP-hard (Garey and Johnson 1979). The integrated planning and scheduling problem dealt with here is harder still, involving larger search spaces and more complex constraints, and hence has not attracted much attention until recently. A number of researchers have developed scheduling techniques that allow a small number of options in their process plans (Sycara *et al* 1991, Tonshoff *et al* 1989) , but still they are dealing with only a small fraction of the whole problem. Liang and Dutta (1990) have pointed out the need to combine planning and scheduling, but their proposed solution was demonstrated on a very small simplified problem. Given a problem of this complexity it is natural to appeal to stochastic optimisation techniques, hence the development of the GA-based method reported here. Comparisons with other techniques are discussed later in section G9.5.4.

The Distributed Co-evolutionary GA

A major early concern in this work was how to provide coherent coevolution. The initial, somewhat unsatisfactory, implementation involved a set of interacting standard sequential GAs and is described in Husbands and Mill (1991). A later, more satisfactory implementation, that has been used ever since, spreads each population 'geographically' over *the same* 2D toroidal grid, this is illustrated in figure G9.5.4. Each cell on the grid contains exactly one member of each population. Selection is local, individuals can mate only with those members of their own species in their local neighbourhood. Following Hillis (1990) the neighbourhood is defined in terms of a Gaussian distribution over distance from the individual; the standard deviation is chosen so as to result in a small number of individuals per neighbourhood. Neighbourhoods overlap allowing information flow through the whole population without the need for global control. Selection works by using a simple ranking scheme within a neighbourhood: the most fit individual is twice as likely to be selected as the median individual. Offspring produced replace individuals from their parents' neighbourhood. Replacement is probabilistic using the inverse scheme to selection. In this way genetic material remains spatially local and a robust and coherent coevolution (particularly between Arbitrators and process plan organisms) is allowed to unfold. Interactions are also local: costing involves the simulation of the concurrent execution of all the plans *at the same location on the grid* (there will be one for each component, and an Arbitrator to resolve conflicts). This implementation consistently gives better results in fewer evaluations than the first. For full details see Husbands (1993, 1994).

The overall algorithm is quite straightforward. It can be implemented sequentially or in a parallel asynchronous manner, depending on available hardware.

```
Overall()
(i) Randomly generate each population, put one member of each population in each cell of a
    toroidal grid.
(ii) Cost each member of each plan population (phase1 + phase2 costs). Phase 1 costs are those
    intrinsic to a given plan (basic machining costs). Phase2 costs include waiting times and are
    calculated by simulating the concurrent execution of all plans represented in a given cell on
    grid, any resource conflicts are resolved by Arbitrator in that cell. Cost Arbitrators according
    to how well conflicts resolved.
(iii) i ← 0.
(iv) Pick random starting cell on the toroidal grid.
(v) Breed each of the representatives of the different populations found in that cell.
(vi) If all cells on the grid have been visited Go to (vii). Else move to next cell,Go to (v).
(vii) If $i <$ MaxIterations, i ← i + 1, Go to (iV). Else Go to (viii).
(viii) Exit.
```

The breeding algorithm, which is applied in turn to the members of the different populations, is a little more complicated.

```
Breed(current_cell,current_population)
(i) i ← 0.
(ii) Clear NeighbourArray
(iii) Pick a cell in neighbourhood of current_cell by generating x and y distances (from current_cell)
    according to a binomial approximation to a Gaussian distribution. The sign of the distance
    (up or down, left or right) is chosen randomly (50/50).
(iv) If the cell chosen is not in NeighbourArray, put it in NeighbourArray, i ← i+1, Go to (v). Else
    Go to (iii).
(v) If $i <$ LocalSelectionSize, Go to (iii). Else Go to (vi).
(vi) Rank (sort) the members of current_population located in the cells recorded in NeighbourArray
    according to their cost. Choose one of these using a linear selection function.
(vii) Produce offspring using the individual chosen in (vi) and current_population member in
    current_cell as the parents.
(viii) Choose a cell from ranked NeighbourArray according to an inverse linear selection function.
    Replace member of current_population in this cell with offspring produced in (vii).
(ix) Find phase one (local) costs for this new individual (not necessary for Arbitrators).
(x) Calculate new phase two costs for all individuals in the cell the new individual has been placed
    in, by simulating their concurrent execution. Update costs accordingly.
(xi) Exit.
```

The binomial approximation to a Gaussian distribution used in step (iii), falls off sharply for distances greater than 2 cells, and is truncated to zero for distances greater than four cells.

Requirements

The architecture of the evolutionary systems is such that the evaluation functions can easily be changed to meet the particular requirements of a specific application of the general model. However, the overall requirements will always be the same: minimise the cost of the manufacturing plan for each component (according to particular criteria chosen, e.g. machining and setup costs) and *at the same time* minimise some higher-level criteria such as makespan, mean flowtime, total tardiness, or some combination of these (French 1982).

Representation

As already mentioned, there have been a number of applications of the ecosystems model to different integrated manufacturing planning problems. Each of these has used the same encoding scheme for the Arbitrators, but the process plan encodings have been tailored to the particular instance of the integrated problem. The encoding scheme used in the case study reported here will be the only one described in this paper; for a more complex encoding used for a very general version of the problem see Husbands (1993).

For this instance of the problem the process plan chromosomes are divided into two sections:

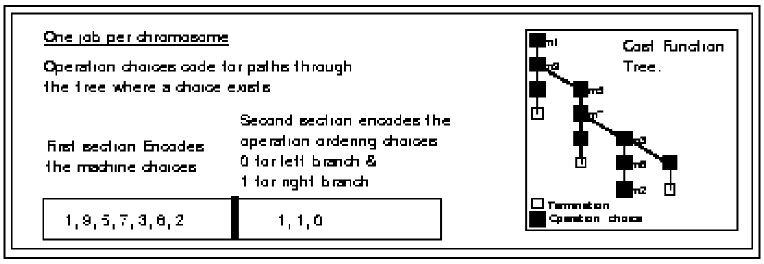
One job per chromosome
Operation choices code for paths through the tree where a choice exists
First section Encodes the machine choices
Second section encodes the operation ordering choices 0 for left branch & 1 for right branch
1,9,5,7,3,6,2
1,1,0
Cost Function Tree.
m1
m9
m5
m7
m3
m6
m2
Termination
Operation choice

Figure G9.5.5. Process plan encoding

the first part deals with method (i.e. machine) choices, the second with sequence (or ordering) choices, see figure G9.5.5. Method choices are only denoted for jobs where there is more than one applicable method. Currently, all methods have two options and are therefore represented as bits in a bitstring. Lookup tables in the cost function translate these binary values into a machine choice. The method choices are held on the genome in an order which maps on to a set of known operations (1-N), which can be considered the default sequence. For each job, the cost function (see later) maintains a data tree containing the space of legal sequences of operations. Sequence choices on the chromosome are interpreted as routes down the sequence tree for a particular job. The default sequence is always legal, so in cases where the problem description constrains the genome to only one legal sequence, the sequencing information is implicit. The evaluation function is a set of 'data abstraction' routines that traverse a given tree structure, following a route taken as argument, which return with a necessarily valid operation sequence.

The Arbitrators are required to resolve conflicts arising when members of the other populations demand the same resources during overlapping time intervals. The Arbitrators' genotype is a bit string which encodes a table indicating which population should have precedence at any particular stage of the execution of a plan, should a conflict over a shared resource occur. A conflict at stage L between populations K and J is resolved by looking up the appropriate entry in the Lth table. Since population members cannot conflict with themselves, and we only need a single entry for each possible population *pairing*, the table at each stage only needs to be of size $N(N-1)/2$, where N is the number of separate component populations. As the Arbitrators represent such a set of tables flattened out into a string, their genome is a bit string of length $SN(N-1)/2$, where S is the maximum possible number of stages in a plan. Each bit is uniquely identified with a particular population pairing and is interpreted according to the function given in Equation G9.5.5.

$$f(n_1, n_2, k) = g\left[\frac{kN(N-1)}{2} + n_1(N-1) - \frac{n_1(n_1+1)}{2} + n_2 - 1\right] \tag{G9.5.5}$$

Where n_1 and n_2 are unique labels for particular populations, $n_1 < n_2$, k refers to the stage of the plan and $g[i]$ refers to the value of the ith gene on the Arbitrator genome. If $f(n_1, n_2, k) = 1$ then n_1 dominates, else n_2 dominates. By using pairwise filtering the Arbitrator can be used to resolve conflicts between any number of different species.

Evaluation Functions

Each job, j, has the following data associated with it: release date r_j; due date d_j; completion time C_j; flowtime $F_j = C_j - r_j$; lateness $L_j = C_j - d_j$; tardiness $T_j = max(0, L_j)$; processing time of job j on machine i, P_{ij}.

From this data the following kinds of cost functions can be calculated in a straightforward manner. makespan: C_{max}; mean flowtime: $\frac{1}{N}\sum_{j=1}^{N} F_j$; total tardiness: $\sum_{j=1}^{N} T_j$; proportion of tardy jobs.

A number of different evaluation functions were experimented with. Particularly good results were obtained with the objective function, O, shown in equation G9.5.6. This function is to be minimised.

$$O = \frac{1}{N}\sum_{j=1}^{N} F_j + 2 \times \sum_{j=1}^{N} T_j \qquad \text{(G9.5.6)}$$

This function, mean flowtime plus twice the total tardiness, is applied to each member of each cell on the 2D grid, including the Arbitrators. The flowtime term encourages individually efficient plans and the tardiness term encourages minimal interactions between the plans.

G9.5.3 Development and Implementation

The system was developed in C under Unix running on Sun workstations. The distributed coevolutionary GA makes use of the MPI parallel message passing interface protocol, allowing it to run on single workstations, networks of workstations and specialised parallel machines.

G9.5.4 Results

This section presents results from runs on 100 problems generated from data provided in Palmer (1994). Table G9.5.1 gives the values for various criteria averaged over the 100 problems. The distributed coevolutionary GA (CDGA) results are shown alongside those previously found by Palmer with simulated annealing (SA) and local dispatching rule heuristics (K&C).

Algorithm	makespan	proportion tardy	total tardiness	totalmachining time	mean flowtime
CDGA	81.22	0.14	5.84	171.75	34.86
SA	89.09	0.18	8.87	191.22	36.10
K&C	95.96	0.31	30.28	218.13	41.37

Table G9.5.1. Problem set comparison

As can be seen from table G9.5.1 the distributed coevolutionary GA outperforms SA and K&C on all of the optimisation criteria. The mean improvement over SA, averaged over all of the optimisation criteria is 16.58%. The mean improvement over K&C, averaged over all of the optimisation criteria, is 37.60%. Each of the methods was run for a comparable number of evaluation function calls.

G9.5.5 Conclusions

In this case study of a complex manufacturing planning problem, we found that for each of a wide range of optimisation criteria the ecosystems model consistently outperformed simulated annealing and a dispatching rule algorithm. Unlike any of the other techniques, the coevolutionary distributed GA produces a number of unique (and quite different) high quality solutions to the problem on each run. Typically the CDGA would generate eight or nine unique very high quality solutions to a given problem on a single run. This work has involved adapting Husbands' coevolutionary model of integrated production planning for use with a new set of problems and with different cost functions to those used previously (Husbands 1993). This adaptation turned out to be relatively straightforward, an experience that supports the claim that the coevolutionary model is very general (Husbands 1993).

Acknowledgements

This work was supported by EPSRC grant GR/J40812.

References

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

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

French S 1982 *Sequencing and scheduling: an introduction to the mathematics of the job-shop* (Ellis Horwood: Chichester)

Garey M and Johnson D and Sethi R 1976 Complexity of flowshop and jobshop scheduling *Math. Opns. Res.* **1**

Garey M and Johnson D 1979 *Computers and intractability: a guide to the theory of NP-Completeness* (W.H.Freeman)

Graves S 1981 A review of production scheduling *Operations Research* **29(4)** 646–667

Hillis W D 1990 Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure *Physica D* **42** 228–234

Husbands P and Mill F 1991 Simulated Co-Evolution as the Mechanism for Emergent Planning and Scheduling *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, ICGA-9* eds. Belew R. and Booker L. (Morgan Kaufmann: San Mateo, CA) pp 264–270

Husbands P 1993 An Ecosystems Model for Integrated Production Planning *Intl. Journal of Computer Integrated Manufacturing* **6(1&2)** 74-86

Husbands P 1994 Distributed Coevolutionary Genetic Algorithms for Multi-Criteria and Multi-Constraint Optimisation *Evolutionary Computing, AISB Workshop Selected Papers, Lecture Notes in Computer Science* **Vol. 865** ed. Fogarty T (Springer-Verlag: Berlin) pp 150–165

Liang M and Dutta S 1990 A mixed-integer programming approach to the machine loading and process planning problem in a process layout environment *Int. Journal Production Research* **28(8)** 1471–1484

McIlhagga M and Ives R and Husbands P 1995 A Comparison of Simulated Annealing, Dispatching Rules and a Co-evolutionary Distributed Genetic Algorithm as Optimisation Techniques for Various Integrated Manufacturing Planning Problems *GAME Project Report 4* School of Cognitive and Computing Sciences, University of Sussex

Muth J and Thompson G 1963 *Industrial Scheduling* (Prentice-Hall)

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

Palmer G 1994 *An Integrated Approach to Manufacturing Planning* PhD. Thesis, School of Engineering, University of Huddersfield

Sycara K and Roth S and Fox M 1991 Resource allocation in distributed factory scheduling *IEEE Expert* **Feb. 1991** 29–40

Tonshoff H and Beckendorff U and Anders N 1989 FLEXPLAN - A concept for intelligent process planning and scheduling *CIRP Int. Workshop on CAPP* (University of Hanover)