The Application of a Distributed Genetic Algorithm to a

1 Introduction

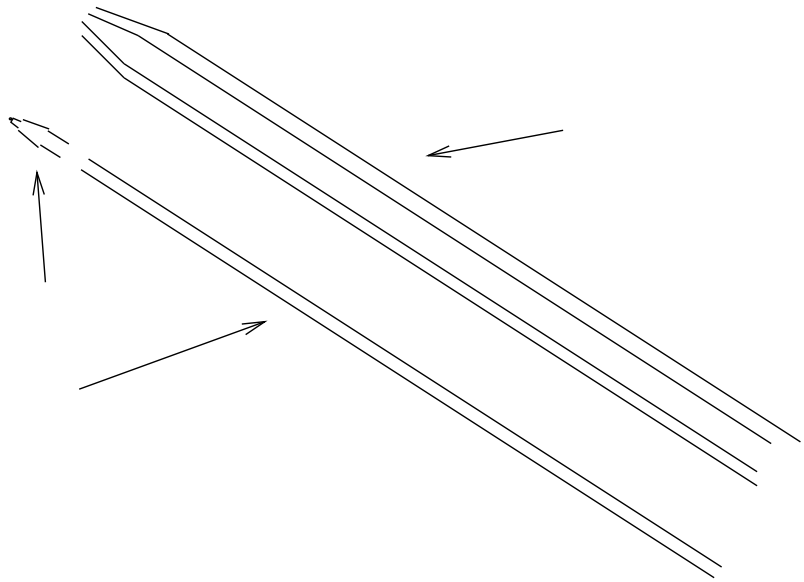
Scheduling in its various guises has been used by the GA community for a number of years to investigate the application of GAs to a challenging and important class of combinatorial optimisation problems. These studies have concentrated on specific scheduling problems or specific classes of scheduling problems [1]; [9]; [6]. This report describes the application of a Distributed GA to *the generic scheduling* problem (i.e. the entire class of scheduling problems). We have achieved this by formulating and implementing a framework for defining, simulating and solving scheduling problems in a generalised way.

The results reported here are based on preliminary testing of the system using 100 large scale problems in a comparison of three scheduling techniques: random search, dispatching rules (a heuristic technique) and a DGA. The problems were generated to reflect the underlying form of JSS that we have tackled previously [6], however they were scaled up to have approximately 50–100 times more schedulable tasks. We found that the random search and dispatching rules methods were able to reduce the makespan of a schedule (using the mean of 10 random solutions as a base for comparison) by about 40 percent. Whereas, the GA was, on average, able to reduce the makespan by 60 percent.

2 A Generic Scheduling System

2.1 Problem Descri tion

MOGS incorporates an SDL that enables the user to describe scheduling problems using a set of



many of the necessary constraints in the problem.

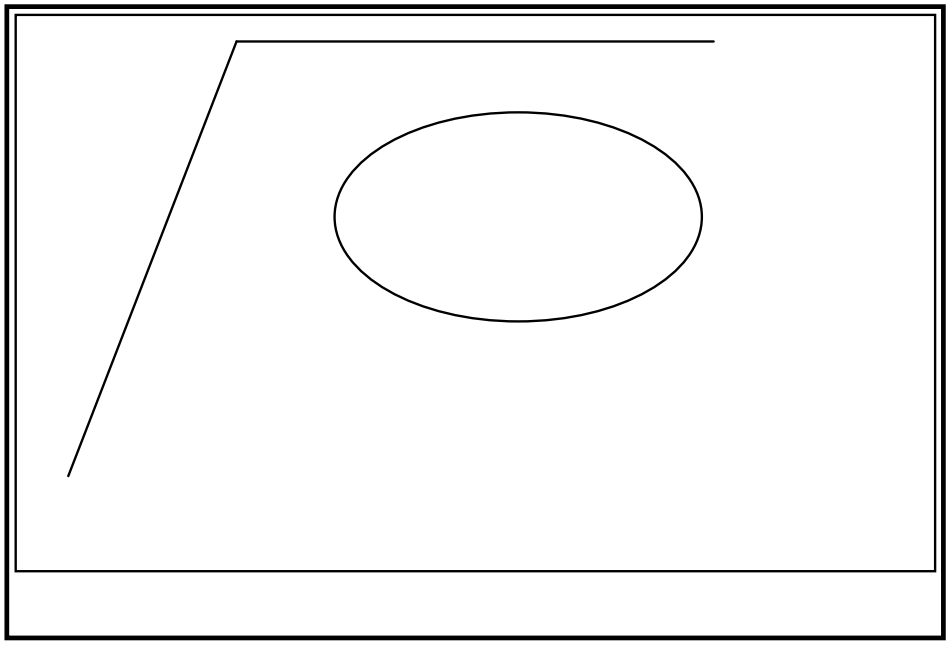
This aspect of the system adds to the dimensionality of the problem. Although the problems we have used for this study are based on job-shop problems they also include the aspect of material supply: transport rate, stock control etc. This more accurately models the real scheduling problems that industry encounters. These problem are in the class of JSS problems, but are much harder than those usually tackled [8] because they have been reformulated to include material scheduling.

2.2 Objective Functions

SMOGS reads the problem description and creates a model of the scheduling environment. Candidate solutions are created by one of the search techniques available, presently: random search, dispatching rules or a DGA. One (or a combination) of a number of possible objective functions are used to determine the worthiness of that schedule. The objective function is a *discrete event simulation* which builds schedules by decoding chromosomes evolved by the DGA. These are mapped into a 'gantt chart' (cf. [2]) like data structure via a 'resource availability graph' (see section 2.3). The resource availability graph is built only once at the beginning of each run. At the present time the user can set the objective function to any one or user defined combination of: makespan, mean flow time, resource utilisation, proportion t(function)Tn.[(reform)104402m0Tdjsrmakjedi[(i2c-14999.7(nejectiv)91000.0,)]li[([2c-14

release date	□□□
□□□□□□□□□□	□□□
□□□□□□□□□□□□□□□□□□	□□□
□□□□□□□□□□	□□□□□□□□□□□□□□□
□□□□□□□□□	□□□□□□□□□□□□□□□

Possible paths through which are defined by each legal chromosome. The tree limits



order and a standard rank selection is applied to choose the second parent.

•

CHROMOSOME STRUCTURE																									
for at	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	GG	GG	G	G	G	GGG	G
type	O	O	O	O	O	O	O	O	T	T	T	T	T	T	T	T	T	TT	TT	T	T	T	TTT	T
ap	N	N	N	N	N	N	N	N																
label	task order								resources								reservoirs							
nu ber																								
bytes																								

Table 2: Example Encoding of a Chromosome.

1. The first T genes contain integers in the range $0 < Tn < T$ denoting the ordering of tasks, i.e. how they are place on the gantt chart. The ordering not only defines the precedence of one task over another on a given resource, but also assigns precedence of material supply to tasks earlier in the order, (even though their time slot might be later than a task later in the order which is placed on a less utilised resource).

2. The next T genes in the chromosome denote the resources that tasks will use to complete. The format is such that the first resource is allocated to task 1 in the problem data structure, not the first task in the ordering defined in the first part of the chromosome.

3. The rest of the chromosome is dedicated to mapping which reservoir will supply what material to which task–reservoir pair. A reservoir can be thought of as a buffer that releases materials to (in the case of a job-shop, machines) the resources which process tasks. It is mapped out in the following way:

   for each task in the problem data structure
   and for each material required by that task
   there is a gene that denotes the reservoir
   that will supply the material.

Example 1: This very simple problem has 8 tasks, 1 of which requires 3 materials, 2 of which requires 2 materials, the rest requiring only 1. The chromosome structure would look like table 2.

ecute. Much of the schedule building work has been 'taken out of the loop' by building the resource availability graph and the chromosome 'range' map at the start of the run. Thus enabling the solution of larger problems. The real significance of this rather specific encoding is that it is not specific to any sub-class of *the generic scheduling problem*, but encodes for the whole class of problems.

5 Operators

Crossover in the first (type O) section of the chromosome works in the following way: a sub-string of one parent is found and inserted into the other parent once the items in the sub-string have been removed from the receiving parent. Sub-string length, position and insertion point are chosen at random.

Crossover for type T is more straight forward. It is implemented in the following way. N crossover points are chosen in the parent chromosomes. Often this will be one crossover point per chromosome, however this can be defined by the user. The section before the crossover point in parent A is concatenated with the section after the crossover point in parent B to form the new child chromosome.

Mutating the first section of the chromosome, which represent a unique ordering of tasks, is more problematic than bit mutation of the binary strings used for the second and third section of the chromosome. Mutation of an ordered set can take a number of forms. In all cases, the restriction that one of each of the numbers in the range $\{1, T\}$, where T is the number of genes in the ordered section of the chromosome, must hold. This can be achieved by implementing 1 or all of:

1. Swapping the order of two juxtaposed tasks in the chromosome.
2. Swapping the order of two task allocated the same resource.
3. Moving a higher priority task on Resource N to just after the juxtapose (lower priority) task on Resource N.

Mutation of type T genes can be done in two ways:

1. Mutate one or more bits. This can cause big or small changes in what any giinm in1.

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□				
	□□□□	□□□□□□□□□□	□□□	□□□
□□□□□□	39.2580	14.0824	22.8734	62.4510

produced by the traditional techniques in place in industry today. This problem set has really stretched the limit of what is possible with current GA technology.

Future work will include a full implementation of the MOGS system design. Although this report has not covered the difference between the MOGS specification and the SMOGS implementation, various classes of scheduling problems can be easily specified using MOGS (which supports a hierarchical task structure) that are difficult to represent and solve using SMOGS. A full implementation of MOGS will also provide a faster and more effective problem modeller.

References

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

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

[3] B. Khoshnevis and Q. Chen. Integration of process planning and schedul-

[9] H. Tamaki and Y. Nishikawa. Parallelled genetic algorithm based on a neighborhood model and its application to job-shop scheduling. In *Proceedings of PPSN II*. Springer Verlag, 1992.