INT. J. COMPUTER INTEGRATED MANUFACTURING, 1993, VOL. 6, NOS. 1 & 2, 74–86

An ecosystems model for integrated production planning

PHILIP HUSBANDS

Abstract. This paper re-evaluates the job-shop scheduling problem by showing how the standard definition is far more restrictive than necessary and by presenting a new technique capable of tackling a highly generalized version of the problem. This technique is based on a massively parallel distributed genetic algorithm and is capable of simultaneously optimizing the process plans of a number of different components, at the same time a near-optimal schedule emerges. Underlying the evolutionary machinery is a specialized feature-based generative process planner.

1. Introduction

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 paper will argue that the standard definition is far more restrictive than is necessary. In particular, it is claimed that the relationship between process planning and scheduling has been largely ignored. A new technique, capable of tackling a highly generalized JSS, is presented. The algorithm used is highly parallel and makes use of methods analogous to those occurring in a natural evolving ecosystem. Underlying the evolutionary machinery is a specialized feature-based generative process planner. It is shown how the technique provides a highly integrated production planning system, treating process planning and scheduling as inextricably interwoven parts of the same problem.

The very large body of work on solving planning and scheduling problems has emanated mainly from the fields of artificial intelligence and operations research. Traditional AI approaches have had limited success in real-world applications, indeed their shortcomings have been thoroughly explored and documented (Chapman 1985). The general resource planning, or scheduling, problem is well known to be NP-complete (Garey and Johnson 1979). Consequently OR techniques have been developed to give exact solutions to restricted versions of the problem, but in general, as with AI-based approaches to the problem, there is a reliance on heuristic-based methods. Because of the complexity and size of the search spaces involved, 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. In many instances this has led to the most general underlying optimization problem being ignored or, more often, not even acknowledged as existing at all.

The most sweeping of these simplifications involves the relationship between process planning and scheduling. Scheduling is traditionally seen as the task of finding an optimal way of interleaving a number of fixed plans which are to be executed concurrently and which must share resources. The implicit assumption is that once planning has finished scheduling takes over. In fact there are often many possible choices for the sub-operations in the plans. Very often the real optimization problem is to optimize simultaneously all the individual plans and the overall schedule. This paper describes how manufacturing planning has been radically recast to allow solutions to the simultaneous plan and schedule optimization problem, a problem previously considered too hard to tackle at all. A model based on simulated co-evolution is described and it is shown how complex interactions are handled in an emergent way. Results from an implementation on a parallel machine are reported. The potential economic benefits are obvious.

The following section makes clear the domain definitions used in the work described. The core techniques used in this research are a specialized form of feature-based process planning and a distributed genetic algorithm. Section 3 gives a brief introduction to genetic

Author: P. Husbands, School of Cognitive and Computing Sciences, University of Sussex, Falmer, Brighton BN1 9QH, UK.

0951-192X/93 $3.00

algorithms and then Section 4 provides an overview of the whole ecosystems model. Section 5 details the feature-based process planning elements of the model, while Sections 6 and 7 describe the parallel genetic algorithm parts. Section 8 presents results from a massively parallel implementation and Section 9 concludes the paper with a discussion of the implications of the work.

2. Domain of application definitions

2.1. *Process planning*

The technique presented here is generally applicable but will be described in terms of the manufacture of medium-complexity prismatic parts requiring the application of a number of metal-removal processes. Within this framework a standard definition of process planning is used (Chang and Wysk 1985), namely establishing the operations required to manufacture a part, the appropriate machine tool and machining parameters to use for each operation and the order in which the operations should be performed. It will be seen that each operation in the plan corresponds to processing a manufacturing feature or group of features on the work piece. Hence, in this work, a process plan is essentially regarded as an ordered set of [feature, machine, process, tool, setup] tuples. Section 5 gives details of the various feature types used and how feature interactions are dealt with.

2.2. *The classical definition of JSS*

The standard JSS problem definition is 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 lengths and involve different subsets of the m machines. The JSS problem is to find the sequence of jobs on each machine in order to minimize 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 (Christophedes 1979).

2.3. *An integrated view of process planning and JSS*

Very often complete fixed process plans are presented as the raw data for the scheduler. However, in many manufacturing environments there is 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 given the machine and tool choices. 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 bottlenecks. Because of the complexity of the overall optimization problem, that is simultaneously optimizing 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. 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 tiny fraction of the whole problem. Liang and Dutta (1990) have also pointed out the need to combine planning and scheduling, but their proposed solution was demonstrated on a very small simplified problem. It is not at all clear if it will scale up to be able to deal with the kinds of test problems described later. The technique presented in this paper, developed by viewing the problem in a completely new way, appears to be the only piece of work fully addressing this highly generalized version of the JSS problem.

3. An introduction to genetic algorithms

Genetic algorithms (GAs) are a key technique used in this work. Since knowledge of the method has not yet spread to all scientific and technical quarters, a brief introduction is given here. For further details see Goldberg (1989), Davis (1990), and Husbands (1992).

We are the existing proof of the astonishing power of natural evolution, a process of selection acting on small variations within a species. It is tempting to imagine that highly effective techniques for optimization, and for the design of adaptive systems, can be abstracted from the logic of natural evolution. Over the past 40 or so years a number of researchers have tried to do just that. The most powerful and successful methods emerged in the late 1960s and early 1970s and are based on Holland's genetic algorithm (Holland 1975).

Genetic algorithms are adaptive search strategies based on a highly abstract model of biological evolution. They can be used as an optimization tool or as the basis of more general adaptive systems. The fundamental idea is as follows. A population of structures, representing candidate solutions to the problem at hand, is produced. Each member of the population is evaluated according to

some fitness function. Fitness is equated with goodness of solution. Members of the population are selectively interbred in pairs to produce new candidate solutions. The fitter a member of the population is the more likely it is to produce offspring. Genetic operators are used to facilitate the breeding; i.e. operators that result in offspring inheriting properties from both parents (sexual reproduction). The offspring are evaluated and placed in the population, quite possibly replacing weaker members of the last generation. The process repeats to form the next generation. This form of selective breeding quickly results in those properties that promote greater fitness being transmitted throughout the population: better and better solutions appear. Normally some form of random mutation is also used to allow further variation. A simple form of this algorithm is illustrated in Figure 1. This population-based survival-of-the-fittest scheme has been shown to act as a powerful problem-solving method over a wide range of complex domains (Grefenstette 1985, 1987, Schaffer 1989, Belew and Booker 1991, Schwefel and Manner 1991, Davis 1990).

The population of structures to undergo adaptation generally consists of strings (chromosomes) of a fixed length. Each element (gene) of the string represents some aspect of the solution and will have a set of possible values (alleles) mapped to various attributes. The fitness of such a string is measured by some objective function that costs the particular combination of attributes present. Hence the chromosomes may be, for instance, strings of real numbers, strings of integers, bit strings (string of 1s and 0s to be decoded into a set of parameter values), a permutation of some set of elements, a list of rules or some combination of these representations.

The set of genetic operators developed by Holland, and the one generally used (possibly with domain-specific modifications), consists of three operators: crossover, inversion and mutation. Simple crossover involves choosing at random a crossover point (some position along the string) for two mating chromosomes, then two new strings are created by swapping over the sections lying after the crossover point. Multi-point crossovers are also frequently used. Inversion is simply a matter of reversing a randomly chosen section of a single string. Mutation changes the value of a gene to some other possible value. The genetic operators are applied at the breeding stage according to a routine like the following. When two strings are selected for breeding, first apply crossover (with some high probability) and randomly choose one of the two new strings thus formed. Next apply inversion (with a medium probability) to this string. Each gene on the resulting string undergoes mutation (with a very low probability) and the outcome is taken as the offspring. The basic operators and the breeding process are illustrated in Figure 2. Note the

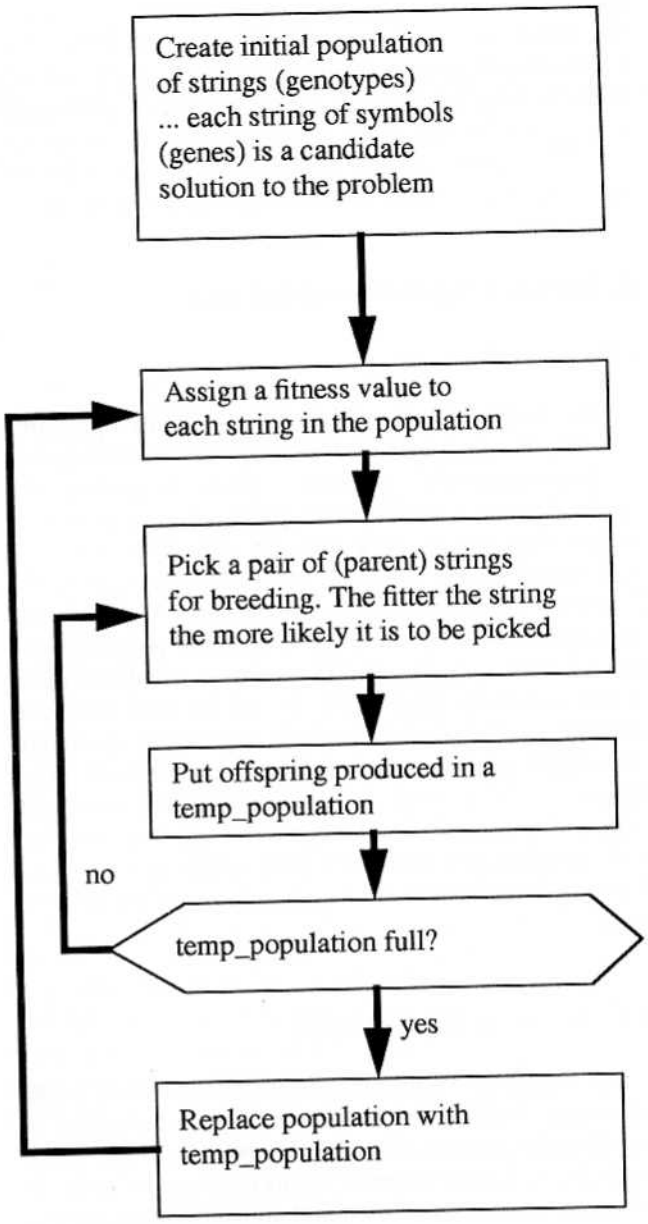

Figure 1. A simple genetic algorithm.

stochastic nature of this process. All operators are applied probabilistically and crossover and inversion points are chosen randomly.

The overall effect is to emphasize combinations of basic building blocks (groups of genes) that produce maximum fitness.

In some problem domains it may be beneficial to allow dynamic length strings. This can be achieved by randomly selecting different crossover points on each parent rather than forcing them to be the same, although recent arguments (Harvey 1992) strongly suggest that changes in length should be restricted to be small and gradual.

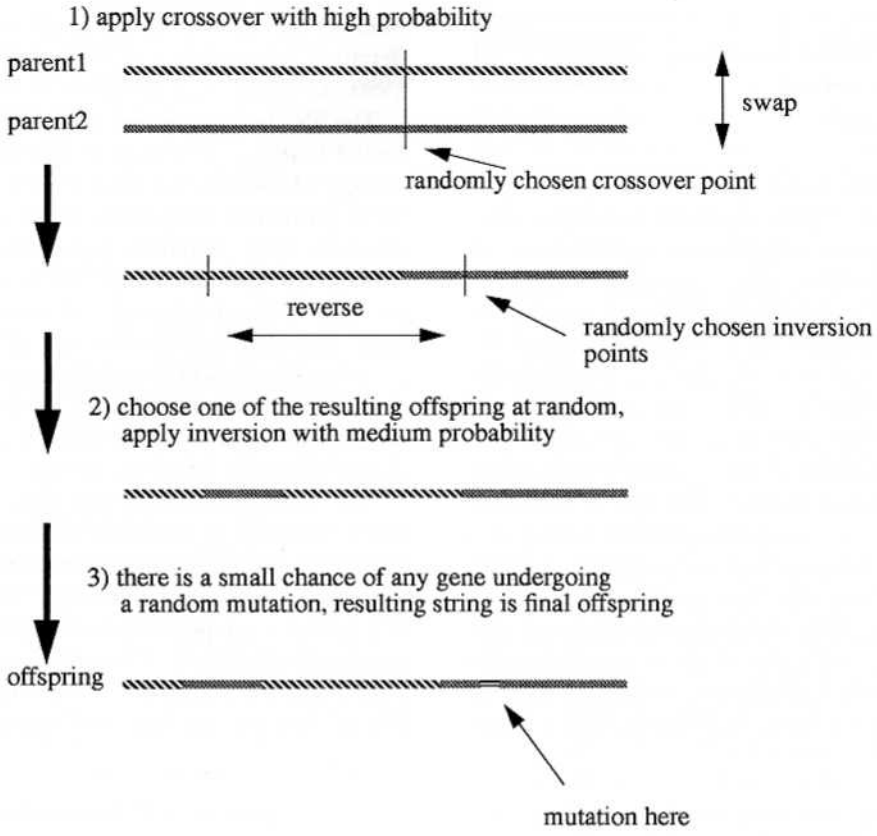

Figure 2. Application of the genetic operators.

Other operators, such as translocation (moving a section of the string to a new location), may also be useful.

There are many variations on and extensions to the basic algorithm. In particular, highly parallel implementations of GAs, with 'geographically' distributed populations and local selection only, appear to be the most powerful (Husbands 1992).

Because Holland and his students developed GAs to serve as adaptive problem-solving strategies able to operate over a large range of environments, their GAs have qualities that make them suitable for many large combinatorial problems and string-representable search tasks. By a combination of selection and reproduction via genetic operators, they are able to find very fit structures by searching only a tiny proportion of the whole problem space. As long as the string representations and the cost function are accurate, GAs can conduct a successful search without recourse to any special domain-specific heuristics. The subtlety of their action prevents them from getting stuck on local optima and ensures that they simultaneously search widely separated parts of the problem space. This is largely due to the random elements in the action of the genetic operators. No assumptions need to be made about the search space, often in contrast to the situation with branch and bound and various heuristic search techniques. Because GAs manipulate populations of legal solutions, they do not suffer from exponential memory usage like many versions of branch and bound and dynamic programming, which attempt to build up gradually a single optimal solution. These qualities make GAs an extremely robust problem-solving method. It is this robustness that makes them an attractive and useful search technique.

Although the basic algorithm is computationally trivial, it should be noted that a great deal of ingenuity is often needed to derive a suitable encoding for a problem and to provide it with an appropriate set of genetic operators and a sufficiently discriminating fitness function. This point will be illustrated later in this paper when the somewhat more complex GA used in this work is described.

4. Overview of ecosystems model

This paper concentrates on two core aspects of a complete framework for dealing with a certain class of design and manufacturing problems. The overall approach is now briefly presented. This is captured, at a very high level, in Figure 3. A design system, whose description is