# Viewing Scheduling Problems through Genetic and Evolutionary Algorithms

Miguel Rocha, Carla Vilela, Paulo Cortez, and José Neves

Dep.Informática - Universidade do Minho - Braga - PORTUGAL
{mrocha,cvilela, pcortez,jneves}@di.uminho.pt

**Abstract.** In every system, where the resources to be allocated to a given set of tasks are limited, one is faced with scheduling problems, that heavily constrain the enterprise's productivity. The scheduling tasks are typically very complex, and although there has been a growing flow of work in the area, the solutions are not yet at the desired level of quality and efficiency. The *Genetic and Evolutionary Algorithms (GEAs)* offer, in this scenario, a promising approach to problem solving, considering the good results obtained so far in complex combinatorial optimization problems. The goal of this work is, therefore, to apply *GEAs* to the scheduling processes, giving a special attention to indirect representations of the data. One will consider the case of the *Job Shop Scheduling Problem*, the most challenging and common in industrial environments. A specific application, developed for a *Small and Medium Enterprise*, the *Tipografia Tadinense, Lda*, will be presented.
**Keywords**: Genetic and Evolutionary Algorithms, Job Shop Scheduling.

## 1 Introduction

In every industrial environment one is faced with a diversity of scheduling problems which can be difficult to solve. Once a good solution is found it produces very tangible results, in terms of the way the resources are used to maximize the profits. The scheduling problems are typically NP-Complete, thus not having the warranty of solvability in polynomial time. Indeed, although there has been a steady evolution in the areas of *Artificial Intelligence (AI)* and *Operational Research (OR)* aiming at the development of techniques to give solution to this type of problems, the basic question has not yet been solved.

The *Genetic and Evolutionary Algorithms (GEAs)* mimic the process of natural selection, and have been used to address complex combinatorial optimization problems. Using an evolutionary strategy, the *GEAs* objective is to maximize/minimize an objective function $f : S(R) \mapsto \Re$, where $S$ is the solution's space. *GEAs* belong to a class of processes designated by *black-box*, once they optimize a function using a strategy independent of the problem under consideration; i.e., require little knowledge of the structure of the universe of discourse.

In this work one aims at studying the application of *GEAs* to address the *Job Shop Scheduling Problem (JSSP)*, a common well-known, scheduling task.

This work is part of a larger project on *Genetic and Evolutionary Computation (GEC)*, where the problem of knowledge representation is addressed in terms of indirect representations of the genotype (the genetic constitution of an organism), that are decoded into a solution. This approach allows the same representation, and its genetic operators to be used in distinct problems. The disadvantage relies on the extra work one is forced to at the decoder's level.

This work was structured into three slopes. The former one gives an introduction to *GEAs*, to the scheduling tasks and to the *JSSP*. In the second slope, one devises the way GEAs can be used in this case and present some of the results obtained. Finally, one comes to the conclusions and prospective work.

## 2 Genetic and Evolutionary Algorithms

### 2.1 Basic Concepts

The *Genetic and Evolutionary Algorithms (GEAs)* are stochastic adaptive systems, whose search methods model natural genetic inheritance and the *Darwinian* struggle for survival. *GEAs* have been used successfully in applications involving parameter optimization of unknown, non-smooth and discontinuous functions, where the traditional algorithmic approaches have been failing (e.g., in industrial drawing, scheduling, planning, financial calculation, data mining). In a *GEA* one begins with a set of individuals, possible solutions to the problem, and proceeds towards finding the best solutions. They operate on an evolving population by applying operators modeled in accordance to the natural phenomena of selection and reproduction. Each solution is represented by an individual, whose genetic constitution, its chromosome, is a sequence of genes, taken from a fixed alphabet, with a well-established meaning. A quality measure is defined in terms of a fitness function, which measures the quality of each chromosome. A new population is formed from stochastically best samples of the previous generations and some offspring coming from the application of genetic operators. The process goes on throughout time until a proper goal is reached (Figure 1). Typical genetic operators include crossover, where the genetic information of two or more individuals are combined, to generate one or two new individuals, and mutation where the genetic information of one individual is slightly altered.

### 2.2 GEPE: The Genetic and Evolutionary Programming Environment

In recent years, there has been a remarkable growth on the *Genetic and Evolutionary Computation (GEC)* field. The models, methodologies and techniques that were developed share common features, overlooked in the software design process. In one's approach, the *GEC*'s applications are implemented as reusable software modules, providing an incremental, manageable and user friendly environment for software development for beginners, being also a good tool for *experts*, who can develop complex applications with increased reusability. To

```
Begin
    Population initialization
    Evaluation of the initial population
    WHILE (termination criteria is not met)
        Select the population's ancestors to reproduction
        Create new individuals, through the application of genetic operators
        Evaluate the new individuals (offspring)
        Select the survivors and add the offsprings to the next generation
End
```

**Fig. 1.** Pseudocode for a Genetic and Evolutionary Algorithm

achieve all these capabilities, one subscribes to the Object-Oriented Programming Paradigm, which allows modularity, encapsulation and reusability of software, by using template fields, abstract classes, virtual methods, default behaviors and values, and re-definition of methods and operators. The proposed framework, named *Genetic and Evolutionary Programming Environment (GEPE)*[3] has three major conceptual layers (Figure 2), namely the *Individuals*, the *Populations* and the *GEA* ones. Each of these layers is defined by a class hierarchy, whose root is responsible for defining the basic attributes and methods. The derived classes can augment or redefine program's behaviors.

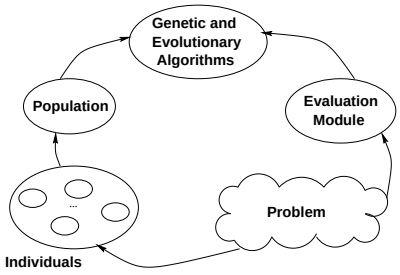


**Fig. 2.** The GEPE Archetype

At the *Individual*'s level one defines the representation and the genetic operators to be used. The genotype of an individual is a set of genes of a given type, implemented as a template, allowing for different types of representational schemes. At the *Population*'s level one defines the initial population creation, the selection and the reinsertion procedures. At the higher level, the *GEA* one, are defined the general structure of the process and the termination criteria. Apart from these three levels, that make the core of the *GEPE*, one needs an extra module to allow for the definition of the problem dependent procedures; i.e., the decoding and the evaluation ones.

## 3 Analysis of the Scheduling Processes

The scheduling problem is concerned with the allocation of scarce resources to tasks throughout time; i.e., it is a decision-making process that produces a plan of procedure for a project, allotting the work to be done and the time for it. The resources and tasks can take varied forms. The resources can be machines in a factory or tracks in an airport. The tasks can be operations in a production process, landings and lift-offs in an airport or executions of computer programs. Each task can have a different level of priority, or a finalization time. The problem may be concerned either with the minimization of the last task finalization time, or of the number of tasks to be completed at an earlier time.

Most part of the scheduling problems can be described in terms of the *Job Shop Scheduling Problem.* Consider an industrial atmosphere where $n$ tasks (operations) will be processed by $m$ machines. Each task is associated with a set of restrictions with respect to the machine's sequence, as well as to the processing time in each one of those machines. The tasks can have different duration times and involve different subsets of machines. One's ordeal aims at an ideal sequence of tasks for each machine in order to minimize a pre-defined objective function, in accordance to the restrictions stating that the order's sequence of processing must be followed, a machine cannot process two (or more) tasks at the same time, and that different tasks for the same order cannot be processed simultaneously by different machines.

More formally, one has a set $J$ of $n$ tasks, a set $M$ of $m$ machines, and a set $O$ of operations. For each operation $op \in O$, there is a task $j_{op} \in J$ to which a machine $m_{op} \in M$ is conjuncted, where task $j_{op}$ will be processed, in a given time $t_{op} \in \Re$. There is also a temporary binary ordering relation that decomposes the set $O$ on a group of partially ordered sets, according to the tasks; i.e., if $x \rightarrow y$ then $j_x \rightarrow j_y$ and there is not a $z$ different from $x$ or $y$, such that $x \rightarrow z$ or $z \rightarrow y$. Electing as objective the minimization of the time elapsed, with the processing of all tasks (making the spanned of the objective function), the problem consists on seeking an initial time $s_{op}$, for each operation $op$, in such a way that the function: $max(s_{op} + t_{op})$ and $op \in O$, is minimized, taking into attention the restrictions (where $t_{op}$ stands for the operation processing time):

**(i)** $t_{op} \in O, \forall op \in O$
**(ii)** $s_x - s_y \geq t_y$ if $y \rightarrow x$, and $x, y \in O$
**(iii)** $(s_i - s_j \geq t_j) \vee (s_j - s_i \geq t_i)$ if $m_i = m_j$, and $i, j \in O$

## 4 Approaching the JSSP with GEAs

A number of the scheduling applications use an indirect representation; i.e., the *GEAs* act on a population of solutions coded for the scheduling problem, and not directly on the schedule. Thus, a transition on such a representation has to be done by a decoder before the evaluation process. This decoder has to seek the information that is not supplied by the chromosome. Its activity will be concerned with the amount of information coded onto the chromosome; i.e., the

least information represented, the more will be done by the decoder, and vice-versa. The domain's knowledge is here used only on the evaluation phase, just to fix the chromosome's fitness (Figure 3).

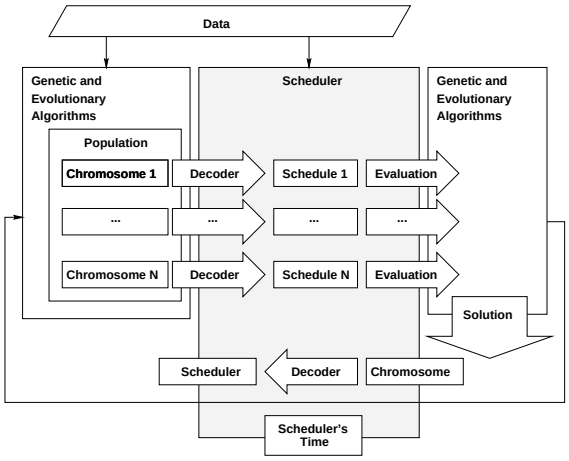


**Fig. 3.** An Indirect Representation

The representation that is used in one's approach is an *Order- Based (OBR)* one; i.e., the genotype is a permutation of symbols, taken form a given alphabet, with no repetitions. In the *JSSP*, the alphabet will be a set of orders to be scheduled, numbered sequentially. Each chromosome will then be a list of orders (Figure 4). Under this setting, the arrangement of elements on the list represent the order's priorities on the schedule. The scheduling problem is reduced to a sequentially ordered set problem, like the *Traveling Salesman Problem (TSP)*. Per each chromosome, the decoder generates the respective production schedule in accordance to the order's sequence - the first order in the list is scheduled in first place, and so forth.

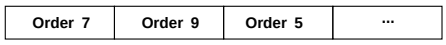


**Fig. 4.** A chromosome for the JSSP

There are a number of genetic operators developed to work with *OBRs* and the following were used in this work:

– *Order Preserving Crossover (OPX)* - *OPX* emphasizes the relative order of the genes from both parents, working by selecting cutting points (1 or 2) and

then building the several segments, in the offspring, by taking genes from alternating parents, maintaining their relative order.
- *Uniform Order Preserving Crossover (UOPX)* - It has some similarities with the previous one, since it maintains the relative order of the genes given by the ancestors. It works with a randomly generated binary mask, that determines which parent is used to fill a given position.
- *Partially Matched Crossover (PMX)* - Under the *PMX* [1] two crossing points are chosen, defining a *matching section*, used to effect a cross between the two parents, through position-to-position exchange operations.
- *Cycle Crossover (CYCX)* - Cycle crossover [4] performs recombination under the constraint that each gene in a certain position must come from one parent or the other.
- *Edge Crossover (EDGX)* - The edge family of crossover operators is based on maintaining all possible pairs of adjacent genes (edges) in the string. It was specially designed for the *TSP* [6].
- *Maximum Preservative Crossover (MPX)* - The *MPX* operator was designed by Mühlenbein [2] with the purpose to tackle the *TSP* by preserving, in the offspring, subtours contained in both parents.
- *Schleuter Crossover (SCHX)* - The *SCHX* [5] is a variation of the previous operator, with some features similar to the *OPX* ones, that also contemplates the process of inversion of partial tours.
- *Adjacent mutation(ADJ)* - It swaps the positions of two adjacent genes.
- *Non-adjacent mutation (NADJ)* - It swaps the positions of two random genes.
- *K-permutation mutation (KPERM)* - It scrambles a sub-list of genes.
- *Inversion(INV)* - It inverts a partial sub-list of genes.

## 5 Results

In this section one describes the results obtained with *GEAs* using different configurations of the genetic operators. In all the experiments were used two operators: a crossover and a mutation one. Table 1 shows the results, for all possible combinations of the operators given above. Each result was computed as the average of 20 independent runs. The instance of the *JSSP* used was generated randomly, considering 5 machines, 2 production plans and 50 orders.

## 6 A practical example

It is one's goal to produce a production scheduling of a typography, such as the Tipografia Tadinense, Lda, a small enterprise located near Oporto, in the north of Portugal. It is intended to produce a solution to give answer to three fundamental questions:

- the priority orders are not handled with the precedence that they deserve;
- there is subutilization of resources (e.g. raw materials, machinery);
- there are too many inactive times due to changes in machine's configuration.

**Table 1.** Results obtained for a JSSP with different genetic operators

| Crossover / Mutation | **ADJ** | **NADJ** | **INV** | **KPERM** |
|---|---|---|---|---|
| **OPX1** | 5019.33 | 1095.41 | 2929.91 | 1607.77 |
| **OPX2** | 3024.11 | 1214.8 | 1554.36 | 1797.5 |
| **UOPX** | 1127.49 | 1133.99 | 1139.51 | 1141.4 |
| **MPX** | 1109.83 | 1116.9 | 1235.17 | 1199.22 |
| **PMX** | 1098.78 | 1165.12 | 1172.46 | 1235.15 |
| **CYCX** | 5956.23 | **1092.33** | 2083.03 | 3022.5 |
| **SCHX** | 1213.54 | 1464.34 | 1358.09 | 1220.9 |
| **EDGX** | 2428.82 | 2956.41 | 2100.06 | 2763.16 |

The intended scheduling is a typical *JSSP* one. Indeed, given a production order, it goes through a set of machines, sequentially It is one's aim to minimize the execution time of an orders portfolio, to give orders the right of precedence over others and to reduce dead times. The evaluation function receives as input a chromosome and computes its fitness. Whenever in it there are non-priority orders, preceding priority ones, it computes a penalty, returning the foreseen time for the scheduling process plus the penalty of the orders portfolio.

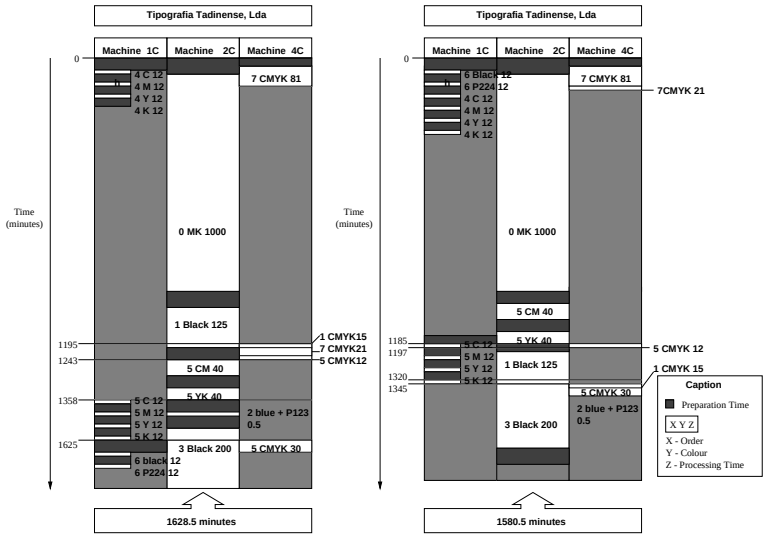


**Fig. 5.** A Snapshot of a Proper Scheduling and a possible solution to the problem via GEAs

A typical order's portfolio, with their associated priorities, the quantities involved, the executing times, the colors and the machines needed was provided.

In Figure 5(left) one can find the schedule used to process it, resulting in a total time of 1628.5 minutes. One applied *GEAs*, considering a population of 100 individuals, and the best solution found was the following list of orders: 6 4 0 7 5 1 3 2. Based on this chromosome, was prepared a scheduling (Figure 5). The time needed to process the orders portfolio is now of 1580.5 minutes.

The scheduling that falls back upon the skills of the practitioner, leads to a deficient use of resources but also to an endemic customer's non-satisfaction. Indeed, machine 2C hangs out during 13 minutes, while in the scheduling made by GEAs there are no hang out times. In this case there is also a reduction of the of machine's preparation times. On the other hand, if two or more orders use the same color, the preparation time for color changeover is zero. Under the *GEA* approach the processing time for the orders portfolio had a decreased of 48 minutes, approximately 4%, which is very important in terms of productivity gains.

## 7 Conclusions

It was presented a programming environment that allows one to address complex scheduling tasks in terms of an evolutionary approach. The ongoing work is being directed to the introduction of direct representations, the dynamic scheduling, by considering events such as orders arriving, machines breaking down, raw materials failing, or orders changing priority, the application of Constraint Logic Programming in order to reduce the solutions space and the development of a Client/Server Scheduling Model with Mobil Agents support.

## References


1. D. Goldberg and R. Lingle. Alleles, Loci and the Traveling Salesman Problem. In J.Grenfenstette, editor, *Proc. of the 1st Intern. Conf. on Genetic Algorithms and their Applications*, Hillsdale, New Jersey, 1985. Lawrence Erlbaum Assoc.
2. H. Mühlenbein. Evolution in time and space - the parallel genetic algorithm. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 316–337. Morgan-Kaufman, 1991.
3. J. Neves, M. Rocha, H. Rodrigues, M. Biscaia, and J. Alves. Adaptive Strategies and the Design of Evolutionary Applications. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO99)*, Orlando, Florida, USA, 1999.
4. I.M. Oliver, D.J. Smith, and J. Holland. A Study of Permutation Crossover Operators on the Travelling Salesman Problem. In J.Grenfenstette, editor, *Proc. of the 2nd Intern. Conf. on Genetic Algorithms and their Applications.* Lawrence Erlbaum Assoc., July 1987.
5. M.G. Schleuter. ASPARAGOS - An Asynchronous Parallel Genetic Optimization Strategy. In J.D.Schafer, editor, *Proc. of the 3rd ICGA*. George-Mason Univ., Morgan Kaufman, 1989.
6. T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley. A Comparison of Genetic Sequencing Algorithms. In R.Belew and L.Booker, editors, *Proc. of the 4th ICGA*, San Diego, July 1991. Morgan-Kaufmann Publishers.