Ruiz-Vanoye, Jorge A.; Díaz-Parra, Ocotlán; Zavala-Díaz, José C.; Ponce-Medellín, Ismael R.
Software Engineering applied to Manufacturing Problems

Available in: http://www.redalyc.org/articulo.oa?id=265219741007
Software Engineering applied to Manufacturing Problems

Jorge A. Ruiz-Vanoye, Ocotlán Díaz-Parra, José C. Zavala-Díaz, Ismael R. Ponce-Medellín
1Universidad Autónoma del Estado de Morelos
2Centro Nacional de Investigación y Desarrollo Tecnológico
jorge@ruizvanoye.com

Abstract. Optimization approaches have traditionally been viewed as tools for solving manufacturing problems, the optimization approach is not suitable for many problems arising in modern manufacturing systems due to their complexity and involvement of qualitative factors. In this paper we use a tool of software engineering applied to manufacturing problems. We use the Heuristics Lab software to determine and analyze the solution obtained for Manufacturing Problems.

Keywords: Manufacturing Problems, Software Engineering.

1 Introduction

Optimization approaches have traditionally been viewed as tools for solving manufacturing problems, the optimization approach is not suitable for many problems arising in modern manufacturing systems due to their complexity and involvement of qualitative factors [1].

Intelligent manufacturing (IM) is a man–machine integrated intelligent system composed by intelligent machine and human experts, which can carry out intelligent activities such as analysis, inference, judgment, conception and decision-making during the process of manufacturing. The cooperation between human and intelligent machine will expand, extend and partially replace the mental work of human experts during the process of manufacturing [2]. At the same time, it will collect, store, perfect, share, inherit and develop the manufacturing intelligence of human experts [3].

In other words, Intelligent Manufacturing means the application of Software Engineering, Artificial Intelligence (AI) and Knowledge-based technologies in general to manufacturing problems. This includes a large number of technologies such as machine learning, intelligent optimization algorithms, data-mining techniques, and intelligent systems modelling. Such technologies have so far proved to be more popular than AI Planning and Scheduling in such applications.

The Job Shop Scheduling Problem (JSSP)[4] contains a number of machines and a set of Jobs each one with precedence restrictions, the problem is to solve the question if exist a scheduling of jobs that help to improve and to efficiency the use of the machines being eliminated the idle times. It is recognized by that it does not have to be able human nor machine sufficiently fast that it can obtain the optimal solution for JSSP due to the solutions space, which cannot be expressed by a polynomial function (deterministic algorithm), the space of solutions for this kind of problem can be only expressed like an exponential function. For the problem of JSSP is necessary to diminish makespan ($c_{\text{max}}$), this can be formulated as it follows (equations 1, 2, 3, 4):

\[
\begin{align*}
\min \ c_{\text{max}} \\
 c_{hk} - t_{jk} &\geq c_{j\ell} \quad j = 1,2,...,n \quad h,k = 1,2,...,m \\
 c_{ik} - c_{ik} + M(1-x_{ik}) &\geq t_{jk} \quad i,j = 1,2,...,n \quad k = 1,2,...,m \\
x_{ik} &\geq 0, x_{ik} = 1 \text{ or } 0, i = 1,2,...,n \quad k=1,2,...,m
\end{align*}
\]

In this paper we use a tool of software engineering applied to solve manufacturing problems. In the section 2 are the software engineering applied to Flexible Manufacturing; in the section 3 are the results of manufacturing problems, and finally the conclusions.

Received March 15, 2010 / Accepted May 10, 2010
2 Software Engineering applied to Flexible Manufacturing

We use the Genetic Algorithm contained in the Heuristics Lab software [5]. A genetic algorithm (GA) is one of the heuristic methods used to find approximate solutions to NP-complete problems. GA is inspired by the Darwinian principles of the evolution of the species, and use own techniques of the genetics, such as: inheritance, mutation, natural selection and recombination (or crossover). The simplest form of genetic algorithm involves three types of operators: selection, crossover (single point), and mutation [6, 7]. Selection, this operator selects chromosomes in the population for reproduction. The fitter the chromosome, the more times it is likely to be selected to reproduce. Crossover, this operator randomly chooses a locus and exchanges the subsequences before and after that locus between two chromosomes to create two offspring. The crossover operator roughly mimics biological recombination between two single-chromosome organisms. Mutation, this operator randomly flips some of the bits in a chromosome. Mutation can occur at each bit position in a string with some probability. The genetic algorithm [8, 9] was used to verify the time and the quality of instances solution with the purpose of determining if the metric generated classify in complexity terms. The input parameters were: selection operator = Roulette, crossover operator = OX, mutation operator = Simple Inversion, generations = 1000, population size = 100, mutation rate = 0.05, replacement strategy = Elitism, crossover rate = 1, n-Elitism = 1, tournament group size = 2.

Many approaches, such as, Simulated Annealing (SA) [10, 11, 12], Tabu Search (TS) [13, 14], Genetic Algorithm (GA) [15, 16], Ant Colony Optimization (ACO) [17, 18], Neural Network (NN) [19, 20], scatter search (SS) [21, 22], particle swarm optimization (PSO) [23, 24] and other heuristic approach [25, 26, 27, 28], have been successfully applied to solve JSSP. In this paper we use the Genetic Algorithm contained in the Heuristics Lab Software.

3 Results

The experimentation was carried on a HP 110-1000 computer with an Intel Atom processor at 1.6 GHz, 1 GB of primary memory. The instances were obtained from the JSSP benchmark [29].

The Table 1 contains the results of applied or uses the Heuristics Lab Software (Genetic Algorithms) on JSSP instances. The la01 instance is more complex than the la05 instance is verified with the quality of the obtained solution. Also sample that in between the yn1 and yn4 instances, the yn4 instance is more complex that yn1 instance (Fig. 1-4). Where: d is the difference between best know and the obtained solution, t is the solution time for the instance on the GA algorithm.

<table>
<thead>
<tr>
<th>Instances</th>
<th>J*M</th>
<th>GA</th>
<th>d</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>abz5</td>
<td>10x10</td>
<td>4.29</td>
<td>0.55</td>
<td>3</td>
</tr>
<tr>
<td>abz6</td>
<td>10x10</td>
<td>3.39</td>
<td>1.09</td>
<td>3</td>
</tr>
<tr>
<td>abz8</td>
<td>20x15</td>
<td>16.6</td>
<td>2.00</td>
<td>8</td>
</tr>
<tr>
<td>abz9</td>
<td>20x15</td>
<td>14.3</td>
<td>1.46</td>
<td>8</td>
</tr>
<tr>
<td>ft06</td>
<td>6x6</td>
<td>8.06</td>
<td>0.08</td>
<td>5</td>
</tr>
<tr>
<td>ft10</td>
<td>10x10</td>
<td>0.00</td>
<td>0.54</td>
<td>4</td>
</tr>
<tr>
<td>ft20</td>
<td>20x5</td>
<td>3.43</td>
<td>0.35</td>
<td>3</td>
</tr>
<tr>
<td>la01</td>
<td>10x5</td>
<td>0.15</td>
<td>0.01</td>
<td>2</td>
</tr>
<tr>
<td>la02</td>
<td>10x5</td>
<td>0.15</td>
<td>0.01</td>
<td>2</td>
</tr>
<tr>
<td>la03</td>
<td>10x5</td>
<td>0.15</td>
<td>0.01</td>
<td>2</td>
</tr>
<tr>
<td>la04</td>
<td>10x5</td>
<td>0.15</td>
<td>0.01</td>
<td>2</td>
</tr>
<tr>
<td>la05</td>
<td>10x5</td>
<td>0.15</td>
<td>0.01</td>
<td>2</td>
</tr>
<tr>
<td>la06</td>
<td>15x5</td>
<td>0.22</td>
<td>0.02</td>
<td>2</td>
</tr>
<tr>
<td>la07</td>
<td>15x5</td>
<td>0.25</td>
<td>0.03</td>
<td>2</td>
</tr>
<tr>
<td>la08</td>
<td>15x5</td>
<td>0.25</td>
<td>0.03</td>
<td>2</td>
</tr>
<tr>
<td>la09</td>
<td>15x5</td>
<td>0.25</td>
<td>0.03</td>
<td>2</td>
</tr>
<tr>
<td>la10</td>
<td>15x5</td>
<td>0.23</td>
<td>0.02</td>
<td>2</td>
</tr>
<tr>
<td>orb01</td>
<td>10x10</td>
<td>5.00</td>
<td>0.28</td>
<td>5</td>
</tr>
<tr>
<td>orb02</td>
<td>10x10</td>
<td>5.96</td>
<td>0.23</td>
<td>5</td>
</tr>
<tr>
<td>orb03</td>
<td>10x10</td>
<td>9.65</td>
<td>0.24</td>
<td>4</td>
</tr>
<tr>
<td>yn1</td>
<td>20x20</td>
<td>0.78</td>
<td>1.15</td>
<td>2</td>
</tr>
<tr>
<td>yn2</td>
<td>20x20</td>
<td>2.78</td>
<td>1.18</td>
<td>2</td>
</tr>
<tr>
<td>yn3</td>
<td>20x20</td>
<td>2.07</td>
<td>1.19</td>
<td>2</td>
</tr>
<tr>
<td>yn4</td>
<td>20x20</td>
<td>0.80</td>
<td>1.25</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1. Results obtained by Heuristics Lab Software for the JSSP instances
The solution (1171.9) of the yn1 (20x20) instance of JSSP is show in the Figure 1.

The solution (1222.47) of the yn2 (20x20) instance of JSSP is show in the Figure 2.

The solution (1131.66) of the yn3 (20x20) instance of JSSP is show in the Figure 3.
The solution (1252.63) of the yn4 (20x20) instance of JSSP is show in the Figure 4.

Figure 4. Solution of the yn4 instance.

4 Conclusions

The heuristic lab software is an excellent tool of software Engineering to solve manufacturing problems. HeuristicLab is an environment for heuristic optimization in general and is not limited to any specific optimization paradigm (like Genetic Algorithms, Tabu Search, Simulated Annealing, or Local Search) or optimization problem. Due to a very high level of abstraction it is possible to implement very different kinds of optimization algorithms and problems for HeuristicLab. To fulfill all its requirements in an intuitive and intelligible way HeuristicLab uses modern programming concepts provided by the Microsoft .NET framework like delegates, events, meta-data, XML, etc.

References


