A Tool for Comparing Resource-Constrained Project Scheduling Problem Algorithms

Alexandru-Liviu Olteanu
Computer Science and Automated Control
Faculty of Engineering, “Lucian Blaga” University of Sibiu
Str. Emil Cioran, Nr.4, Sibiu, 550025
ROMANIA
al.olteanu@gmail.com

Abstract: - There are a multitude of optimization problems that are difficult to solve and for which many algorithms have been developed. One area of these problems is scheduling, where a large range of constraints need to be upheld, making the problems NP-hard. The aim of this paper is to present a tool for comparing different methods for the Resource-Constrained Project Scheduling Problem.

Key-Words: - Resource-Constrained Project Scheduling Problem, RCPSP, Ant Colony Optimization, ACO, Genetic, Beam-search, Scheduling, Simulation

1 Introduction
1.1 Scheduling
Scheduling represents a large category of optimization problems. In essence it refers to the task of assigning a number of jobs to a set of machines in such a way that one or more optimization criteria are met.

Peter Brucker [11] classified these problems based on these factors:
- Job Characteristics
- Machine Environment
- Optimization Criterion

The Job Characteristics refer to the characteristics of each job. A job can be preemptive, meaning it can be stopped before completion and restarted at a later stage. They can have precedence constraints (certain jobs can only be started after others have finished), release times (jobs can have a specific time from project start before which they cannot be scheduled), processing times and deadlines. Also certain cases only allow jobs to be scheduled in batches.

The Machine Environment refers to the number of machines that exist, what jobs they can do, how many at a time they can process and how long it takes. There can be only one machines on which jobs are scheduled (i.e. processes being scheduled on a single processor machine), or there can be multiple ones.

In the case of multiple machines these can be parallel machines (identical processing times), uniform machines (different processing times for similar tasks, but with the same relative performance between them) and unrelated machines (different processing times with no relation between them). Also in this category we have shop problems where jobs are divided into operations which can only be processed by certain types of machines. Some machines can multipurpose (they process more than one type of job) and in the case of multiprocessor tasks several machines work together in order to complete the same job.

The Optimization Criterion aims at minimizing a cost function. There is the bottleneck objective (\(\max f_i\)), the sum objective (\(\sum f_i\)) and the weighted sum objective (\(\sum \omega_i f_i\)).

The cost functions can be the completion time (\(C_i\)), lateness (\(L_i = C_i - d_i\)), tardiness (\(T_i = \max(0, C_i - d_i)\)), earliness (\(E_i = \max(0, d_i - C_i)\)), deviation (\(D_i = |C_i - d_i|\)), square deviation (\(S_i = (C_i - d_i)^2\)) and unit penalty (\(U_i = \begin{cases} 0, & C_i \leq d_i \\ 1, & C_i > d_i \end{cases}\)).

1.2 Complexity
Many scheduling problems, due to the large number of parameters are NP-hard.
In the time complexity theory the P class represents the set of problems that have polynomial complexity.

The NP class (Nondeterministic polynomial class) contains the problems that can be solved by nondeterministic algorithms in polynomial time.

NP-complete problems are problems for which there are no algorithms that can solve them in polynomial time, but a given solution can be tested to be optimal in polynomial time.

NP-hard problems are the hardest to solve. There aren’t known ways to solve them in polynomial time and also solutions can’t be tested in an acceptable time either.

Such problems can only be solved using approximation algorithms, which give close to optimal solutions in polynomial time.
2 Problem Formulation

This paper concentrates on the Resource-Constrained Project Scheduling Problem. This means that we have multi processor tasks with precedence constraints, different processing times and the optimization criterion is the total duration of the project.

The problem contains a set of $n$ jobs $J = \{J_1, J_2, ..., J_n\}$ and one of $m$ machines $M = \{M_1, M_2, ..., M_m\}$.

The times needed by each job in order to complete are $P = \{p_1, p_2, ..., p_n\}$.

Each job $J_i$ has a set of machines $m_i \subseteq M$ that need to work at the same time in order to complete.

There are also precedence constraints where each job $J_i$ can only be scheduled after the set of jobs $Pr_i \subseteq J$ have been completed.

Fig. 1 depicts the precedence tree for a problem with 4 jobs and 2 identical machines. Each node represents a job, with the processing time and resource needs written below it. Two additional nodes have been added representing the start and finish phases. They don’t require resources or time to complete, but are simply used in the algorithm as starting phase, and to determine when the solution has been constructed.

A valid solution (Fig. 4) represents the order in which the jobs are processed such that the precedence constraints are met, and at any given time there are no more machines needed for the jobs being processed than the maximum available.

Fig. 3 shows the solution space for the problem, with one optimal solution highlighted.

Solution: 1-2-4-3.

3 Problem Solution

3.1 General

To solve our problem, I developed a tool in C# 2.0 that can run simulations on the set of benchmarks offered by PSPLib[20]. This application is based on another tool that I used to generate the results in my previous work [1,2].

The application offers 2 functions:

- Generate and view schedules
- Compare the performance over time for multiple methods

3.2 Architecture

The application was built as a distributed system. Currently, the application can work over a LAN, where a server application is set on one station and several client applications are set on others.
The server acts as the main user interface, making it possible to select the simulation parameters and assign each client the task of one simulation repeatedly until all have been completed.

The simulation parameters are:
- Methods with corresponding parameters
- Benchmarks
- Time limit for each simulation
- Number of repeats per simulation

For each combination of one method and one benchmark a preset number of simulations will be run. This can be selected from the interface and is used for the purpose of generating representative results.

To combat the problem of using different machines, a test method is run both on the server and on the clients. This method used is Beam-search and is deterministic, generating the same result in approximately the same amount of time. Comparing the simulation times for the test method on the server and on the clients determines the relative performance between the machines. In consequence, slower machines will be assigned a longer simulation time, while faster machines will be assigned a longer one.

After this point the clients become idle and wait for the server to assign them a task to complete. The server sends a method, a benchmark and the simulation parameters. The client begins the simulation, reports its progress and sends the results back, after which it waits for the server to assign it another simulation.

The methods and benchmarks are only sent if they are missing in their corresponding folders on the client machine.

Before any simulation the first function is to read the benchmark and collect the information from it in the form of a PSP object. This object contains the job information, resource data and the precedence constraints.

![Fig. 8](image)

Each method will interact with the PSP object in order to apply the constraints. This interaction is done through instances of PSPstate which hold partial solutions to the problem. A PSPstate object will keep the state of all the jobs (Not Started, Running, Finished), resource usage, and the current time. As time progresses running jobs will finish and free resources, making available other jobs for the scheduling task.

Each algorithm is derived from an abstract class and only needs to implement one method. This method will contain the entire algorithm for solving the problem.

![Fig. 9](image)

The application detects all implementations from a specific folder on disk and uses them through polymorphism. At this stage I have implemented three methods (Fig. 8), a deterministic approach and two stochastic ones:
- Beam-search Algorithm
- Ant Colony Optimization Algorithm
- Genetic Algorithm

Beam-search is deterministic and for the same problem and set of parameters will generate the same solution. Therefore this method is very good for applications that need the algorithm to bring the same quality of the results each time. However, as it can be seen in the results section, this method does not results as good as the other algorithms.

Ant Colony Optimization is based on the stigmergy principle that ants have been shown to follow and has found applications in many fields [2,3,4,6,7,9,10].

Genetic algorithms belong to another category of search algorithms that apply the principles of evolutionary biology such as selection, recombination, mutation and others. They have also
been very successful in solving a large number of problems [5,8,17].

3.3 Results
Below are pictures highlighting the server applications functions.

The first row tabs are used to select the benchmarks and add and edit the methods. The first picture shows the panel containing information regarding the clients. The next picture is used to select the results to be processed. Each individual result can be represented as a Gantt chart, as shown in Fig. 11. Fig. 12 contains a line chart that contains the performance over time of the previously selected methods and benchmarks. Average values can be shown as well as the maximums, minimums and variance.

Below are some results for the three methods for solving the RCPSP problem. The simulation was done on 20 benchmarks from the j120 set from PSPLib. These problems have 120 jobs that need to be scheduled on 4 resource types. Each simulation of 10 seconds was repeated 50 times and the results were averaged in order to provide representative results. The Y axis shows the solution time, while the X axis the simulation time.

4 Conclusion
The application performed very well, bringing good results for the three implementations for the problem. I was able to find the optimal parameters combinations for each method with ease, and leaving the application to
run a large number of simulations brought representative results.
As future work I intend to implement several other methods, especially hybrid ones between stochastic and deterministic ones.

References: