The Royal Institute of Technology

Artificial Intelligence

Project

An evaluation of Domain-Independent Heuristics when applied to the Job-Shop Scheduling Problem

Authors:
Sara Engardt
Magnus Arvidsson

Johan Haslum Fredin
Simon Westerlind

October 14, 2016

Abstract

This report investigates the use of domain-independent heuristics when solving a Constraint Satisfaction Problem (CSP). These problems quickly becomes complex and creates vast search spaces. The use of a good heuristics can efficiently limit the scope of the problem. Several heuristic functions are evaluated by comparing their performance when solving the classical Job-Shop Scheduling Problem.

Contents

1 Topic

The topic of this project is a short study of how different domain-independent heuristics affect the performance of a constraint satisfaction problem (CSP) solver applied to the Job-Shop Scheduling Problem (JSSP). The $n \times m$ JSSP is a sort of scheduling problem, consisting of n jobs to be performed by m machines in a predefined order in the most efficient way.

Constraint based methods have proven to be a good way to encode and solve scheduling problems. However, representing these problems as inequality constraints creates large search space, resulting in complex problems that are hard to solve exact. One way of handling this is by the use of heuristic functions to utilize more of the information represented in the problem definition to reduce the size of the search space.

In this project we have focused on the classical 10x10 JSSP as defined by Muth and Thompson [ref. industrial]. The problem states that ten jobs are to be performed on ten different machines. These jobs require the use of the machines in sequential order and with varying time span. Even though the 10x10 problem might seem small it has proven to be difficult to solve, or as Yamada and Nakano put it:

"The JSSP is not only NP-hard , but it is one of the worst members in the class. An indication of this is given by the fact that one 10 × 10 problem formulated by Muth and Thompson [...] remained unsolved for over 20 years." (Yamada, 1997)

2 Related Work

The inspiration and formulation of the problem was found in the work done by Yamada and Nakano. They study the different methods that have been used to tackle the JSSP problem and how well they perform on different problem formulations. Furthermore it describes how the problem constraints can be represented graphically form in a constraint graph.

The Job-shop scheduling problem is one that has been widely studied. There are several articles with proposed solutions and methods, a couple of these have been of extra interest for us when working with the problem. The article "Variables and Value Ordeting Heuristics for the Job Shop scheduling problem Constraint satisfaction problem" (Sadeh and Fox, 1995) by Sadeh and Fox studies and compares a wide selection of heuristics. One heuristic they evaluate is the Minimum Remaining Value (MRV) heuristic, which we chose to implement. The authors ~~chooses to~~ measure the performance according to the following three criteria, search efficiency, number of experiments solved and average CPU time. The search efficiency was defined as the number of operations to be scheduled over the number of search states that were explored. The number of experiments solved was a measure of how many solutions were found in under 500 search states each. Lastly CPU time which represent the average time required to find a solution.This study served as a guideline for us on how to compare our heuristics.

The book "Artificial intelligence - a modern approach" (Russel and Norwig, 1995) has a chapter about constraint satisfaction problems which was an important resource during this project. It provided us with funda-

mental theoretic ground, for example how to encode the JSSP as a CSP. It also provided us with suggestions of heuristics for the JSSP, such as the Minimum degree heuristics and the Least-constraining-value heuristic.

2.1 Constraints

The goal of the Job-Shop scheduling problem is to find a solution that minimizes the makespan time, while complying with the predefined constraints. In other words to minimizes the amount of time from the start of the first task to the finish of the last for a given test case. In this case the constraints are the order in which operations are to be carried out and what operations can be performed simultaneously. This can be represented in a constraint Graph as in figure 1 below. The graph consists of two types of edges, representing conjunctive and disjunctive constraint respectively. Figure 1 show what operations are to be carried out in what order and which operations are not allowed to be concurrent.

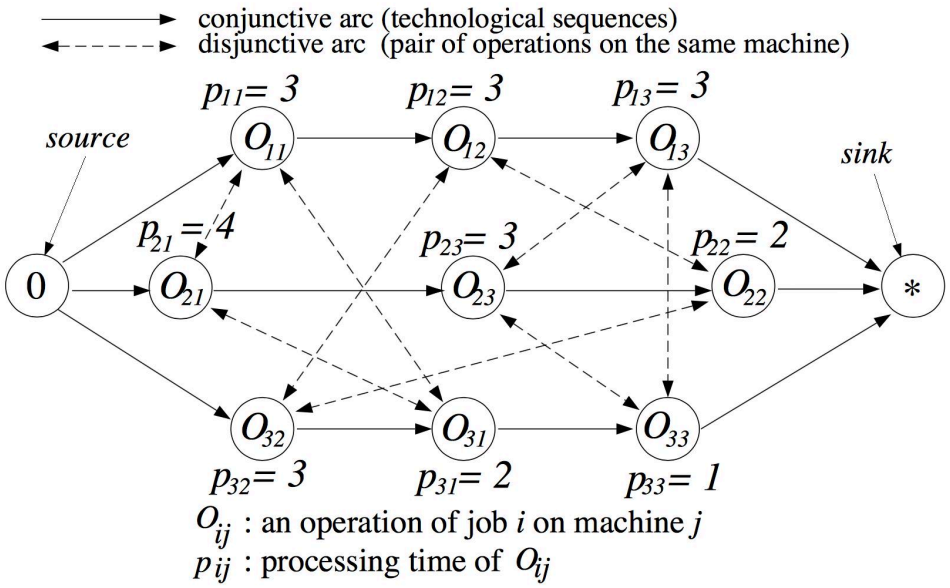

Figure 1: Insert titel

These constraints can also be represented mathematically. To do this, the following notation will have to be introduced. The $n \times m$ Job-Shop problem consists of a set of n jobs J_j $1 \leq j \leq n$, which is to be processed on a set of m machines M_r where $1 \leq r \leq m$. Each job must be processed on a sequence of machines, the processing of job J_j on machine M_r is called the operation O_j^r. During the processing time p_j^r (the time from the start time of the operation plus the duration of the job), the job O_j^r requires exclusive use of machine r. The objective of the JSSP is to find a set of start times for each operation c_j^r $1 \leq j \leq n, 1 \leq r \leq m$, that satisfied these constraints. The time between the start time of the first operation and the end time of the last operation is called the timespan L. When solving the JSSP the aim is find the schedule that minimizes L.

2.2 Constraint Representation

Once again the constraints can be divided into Conjunctive and Disjunctive. Conjunctive constraints simply restrain in what order operations in the same job are allowed to be completed. This can be formulated as requiring the start time of the following operation to be later than the startime of the previous operation and its duration. If operation l follows operation k in job j, the constraint follows:

$$startTime(O_j^k) + p_j^k < startTime(O_j^l)$$

The disjunctive constraints can be derived from the restriction that only one operation can be carried out by a machine at any time, thus requiring two operations competing for the same resource to start when the other has finished. So for operation O_{jr} of job j, requiring machine r and O_{ir} of job i also requiring machine r the constraint can be described as:

$$startTime(O_j^r) + p_j^r < startTime(O_i^r)$$

or

$$startTime(O_i^r) + p_i^r < startTime(O_j^r)$$

2.3 Depth First Branch and Bound

The aforementioned constraints will eliminate many of the possible combinations of Operation orders and limit the space of potential solutions that needs to be searched. This search space is explored with the use of a Depth-First Branch-and-Bound algorithm. The DFBnB algorithm keeps track of the lowest-cost solution found so far. This enables the DFBnB to prune part of the search space, since partial solutions with makespan equal or greater the the lowest-cost solution are unnecessary to explore. Because of this pruning property, a smart choice of which child node is explored can increase the performance of the algorithm significantly. This ranking of child nodes is done with the help of a heuristic functions. This report will explore the different heuristics found below and evaluate their performance.

2.4 Value heuristics

In the JSSP the value corresponds to choosing which of the possible start times[alt. operations*] to schedule. A short description of the value heuristics that we have chosen to implement is given here.

2.4.1 Least Constraining Value

The least constraining value heuristic prioritize the operation which restrain the scheduling of the remaining operations the least. The intention is to cause the least possible conflicts thus quickly finding a possible solution. When implementing this heuristics only the operations incident to the current operation in the constraint graph need to be accounted for, since this operation only affect the connected operations.

2.5 Variable heuristics

In the JSSP choosing a variable corresponds to choosing what operation to perform. A short description of the variable heuristics that we have chosen to implement is given here.

2.5.1 Minimum Remaining Values

The minimum remaining value heuristics also called the "most constrained variable" heuristics prefers variable choices which have the fewest allowed choices left. This "fail-first" approach will be most likely to cause a constraint violation sooner rather than later, thus pruning the tree earlier thereby minimizing the search time (Russel and Norvig, 1995).

2.5.2 Minimum Degree

The Minimum Degree heuristics simply ranks the operations by their degree in the constraint graph. By choosing the operation with the largest degree or in other words the operations that is the most constrained, the most complex operations to schedule are prioritized. This intendeds to create a conflict as early as possible does eliminating large portions of the search space.

2.6 Random choice Heuristic

As a reference point we will compare our heuristics with a complete random choice. This means that we randomly chose the variable and value from the given search space.

3 Case study

3.1 Description

We have used the open source CSP solver "Gecode"[gecode.org] to encode the problem as described in previous sections. We implemented the heuristics in the previous section, except for the random and minimum value heuristic where we used the built-in Gecode functions. We combined the two variable heuristics with the two value heuristics, giving us four test cases plus one reference case of random choices. We tested each combination of heuristics on ten random test cases. When comparing the performance of the heuristics we compare the makespan of the proposed optimal solution, the efficiency in terms of how many nodes in the search tree that was visited for the first and for the optimal and the average time it took to find the first and optimal solution (CPU time). The results are shown in table 1.

3.2 Results

The following table shows the performance of the heuristics.

Heuristics	Shortest makespan	CPU time required to find best solution (μs)	CPU time required to find first solution (μs)	Nodes visited for best solution	Nodes visited for first solution	Nodes failed for best solution
Minimum value and most constraining variable	713	98189	134	681	105	141
Minimum value and minimum degree	709	97795	173	226	55	46
Least constraining value and most constraining variable	845	25205200	223	4973	54	811
Least constraining value and minimum degree	818	30011301	157	4529	50	787
Random choices	811	50618	138	3271	103	947

4 Discussion

We can see blablabla

5 Summary

References

[1] Muth, F. Thompson, L. (1963). Industrial scheduling. Prentice-Hall, Englewood Cliffs, N.J

[2] Sadeh, N. Fox, M. (1995). VARIABLE AND VALUE ORDERING HEURISTICS FOR THE JOB SHOP SCHEDULING CONSTRAINT SATISFACTION PROBLEM. Carnegie Mellon University, Pittsburgh, USA

[3] Yamada, (1997). [online] Available at: `http://www.kecl.ntt.co.jp/as/members/yamada/galbk.pdf` [Accessed 7 Oct. 2016].

[4] Russel, S. Norvig, P. (1995). Artificial Intelligence: A Modern Approach, 3rd ed. pp 216-217. Prentice-Hall, Upper Saddle River, New Jersey, USA

[5] Gecode, (2016). [online] Available at: `Gecode.org` [Accessed 8 Oct. 2016].