A Scalable Method for a Particular Scheduling Problem Based on Boolean Satisfiability

Hai Lin\textsuperscript{1,a}

\textsuperscript{1}College of Software, Shenyang Normal University 253 Huanghe North Street, Shenyang, 110034, China

\textsuperscript{a}jlui_linhai@163.com

Keywords: Boolean satisfiability, scheduling, SAT solving

Abstract. In recent years, people made a lot of progress for solving Boolean satisfiability problem. Existing solvers have been successful in dealing with problems with thousands of Boolean variables in reasonable amount of time. And there is a trend to use Boolean satisfiability solvers to deal with practical problems. In this paper, we investigate a method for reducing a particular type of scheduling problem to Boolean satisfiability. Hence we are able to use existing solvers to solve practical scheduling problems and get good performance.

Introduction

Boolean satisfiability problem is considered one of the most classical problems in computing. Traditionally, it is the first NP-complete problem and has been receiving a lot of attention. But the nature of the problem determines that it is nearly impossible to find any tractable method for solving this problem. However, in the last two decades, people made huge progress in solving Boolean satisfiability problem \cite{1,2,3,4}. Some major techniques involve watched-literal technique \cite{5} and clause learning technique \cite{6}. Due to these advanced techniques, existing Boolean satisfiability solvers are able to perform well on practical problems. These problems may contain thousands of Boolean variables, or even millions in some cases. So there is a trend to use Boolean satisfiability solvers to tackle problems of practical interests.

In this paper, we propose a method, which can reduce a scheduling problem to Boolean satisfiability problem. The idea is as follows. In some process of manufacturing, there are some jobs to be done and some limited resources to use. There are some constraints, which say a particular job can only use some particular resources, but there may be alternative resources. For example, to complete job A, one may either use material M1 and line L1, or use material M2 and line L4. And the goal is to find a way of taking maximum use of available resources to complete all the jobs, subject to the constraints. We will encode the constraints using Boolean formulas. Thus, the scheduling problem can be reduced to a Boolean satisfiability problem.

The rest of the paper is structured as follows. The next section explains precisely what problem we are trying to solve. And then we briefly review Boolean satisfiability problem and methods for solving the problem. In the following section, we explain our method for reducing the scheduling problem to Boolean satisfiability problem. Then we conclude in the last section.

Scheduling Problem

In this paper, we consider the following scenario. Suppose that, in some process of manufacturing, there are m jobs \{J_1, J_2, \ldots, J_m\} and n resources \{R_1, R_2, \ldots, R_n\}. Here resources have a broad sense. They can be human resources, material resources and machine resources. Each job can only use some particular resources. We associate each job \textit{J}_i with a set of sets \{\{R_{k1}, R_{k2}, \ldots, R_{kp}\}, \ldots,\{R_{w1}, R_{w2}, \ldots, R_{wq}\}\}, meaning that the job \textit{J}_i can be accomplished using either resources R_{k1}, R_{k2},\ldots, R_{kp} or resources R_{w1}, R_{w2},\ldots, R_{wq}. For simplicity, we assume that each resource can only be used to complete one job. The goal is to find a way of assigning each job some resources so that all the jobs can be accomplished. Here is a simple example.
Example 1. Suppose that we have 2 jobs \{J_1, J_2\} and 5 resources \{R_1, R_2, R_3, R_4, R_5\}, in which \(R_1\) means the first machine, \(R_2\) means the second machine, \(R_3\) means the first person, \(R_4\) means the second person and \(R_5\) mean the third person. We associate job \(J_1\) with the set \{\{R_1, R_3\}, \{R_1, R_4, R_5\}\}, meaning that job \(J_1\) can be complete using the first machine and the first person, or using the first machine and the second person and the third person. We associate job \(J_2\) with the set \{\{R_2, R_3, R_4\}, \{R_2, R_5\}\}, meaning that job \(J_2\) can be complete using the second machine and the first person and the second person, or using the second machine and the third person. We would like to find a way of assigning each job some resources so that both of them can be accomplished. The solution would be the following. \(J_1\) is assigned \(R_1\) and \(R_3\), \(J_2\) is assigned \(R_2\) and \(R_5\).

**Boolean Satisfiability**

In this section, we review the Boolean satisfiability problem and major techniques for solving the problem. We begin with some basic terminologies. A Boolean variable is a variable that is either true or false. A literal is a Boolean variable of its negation. A clause is the disjunction of a set of literals. A CNF (Conjunctive Normal Form) formula is a conjunction of clauses. For simplicity, we write a clause as a set of literals, and we write a CNF formula a set of sets. Here is an example. \{\{x_1, x_2\}, \{x_1, \neg x_2\}\} simply means \((x_1 \lor x_2) \land (x_1 \lor \neg x_2)\).

The Boolean satisfiability problem can be described as follows. Given a CNF formula, the goal is to find an assignment to each Boolean variable such that each clause in the formula has at least one literal being true. If we can find such an assignment, the formula is satisfiable, otherwise it is unsatisfiable. In the above example, if we assignment true to \(x_1\) and assign false to \(x_2\), the formula is true. So the formula is satisfiable, and \{\(x_1=\text{true}, x_2=\text{false}\)\} is a satisfying assignment.

In order to solve Boolean satisfiability problem, one needs to do an exhaustive search in all possible assignments to all the Boolean variables. If we arrangement all possible assignments in a tree, the nodes in the tree is exponential in the number of Boolean variables. So the naïve search would not be practical. There are some techniques to speed up the search. For example, if we know for sure the value of some Boolean variable, we make that decision as early as possible to make the size of the search tree small. In addition, we use a particular indexing technique, called watched-literal technique, to minimize the number of clause access. Another useful technique is called clause-learning. The idea is as follows. If we get a conflict in some part of the search tree, we try to learn from that failure by deducing a clause, which is a logical consequence of the original formula. The learnt clause will be useful to prevent us from making the same mistake in other parts of the search tree. Due to these techniques, modern Boolean satisfiability solvers are able to handle problems with huge number of Boolean variables.

**Our Method for Scheduling**

In this section, we present our method for reducing the scheduling problem to Boolean satisfiability problem. The idea is as follows. We create some Boolean variables and encode the constraints in the scheduling problem as a Boolean formula using these Boolean variables. The encoding has the property that all the jobs can be accomplished if and only if the Boolean formula is satisfiable and the satisfying assignment corresponds to possible scheduling.

First, we create some Boolean variables of the form \(U_{mn}\), which means that job \(J_m\) uses resource \(R_n\) in the scheduling. According to the constraints, each resource can only be assigned to one job. To encode this constraint, for each different pair of jobs \(J_m\) and \(J_n\), and for each resource \(R_k\), we create the following clause.

\[\neg U_{mk} \lor \neg U_{nk}\] (1)

Intuitively, the above clause means that either job \(J_m\) cannot use resource \(R_k\) or job \(J_n\) cannot use resource \(R_k\). In other words, job \(J_m\) and job \(J_n\) cannot both use resource \(R_k\).

We still have another type of constraint to consider. Remember that each job can only use some
particular recourses. Suppose that we associate a job \( J_i \) with a set of sets \( \{ \{ R_{k1}, R_{k2}, \ldots, R_{kp} \}, \ldots, \{ R_{w1}, R_{w2}, \ldots, R_{wq} \} \} \), meaning that the job \( J_i \) can be accomplished using either resources \( R_{k1}, R_{k2}, \ldots, R_{kp} \) or resources \( R_{w1}, R_{w2}, \ldots, R_{wq} \). Then we will create a formula below.

\[
(U_{ik1} \land U_{ik2} \land \ldots U_{ikp}) \lor \ldots \lor (U_{iw1} \land U_{iw2} \land \ldots U_{iwp}) \quad (2)
\]

Suppose that we take the conjunction of all the formulas corresponding to all the constraints, and get a formula \( F \). \( F \) has the property that all the jobs can be accomplished if and only if the \( F \) is satisfiable.

Let us consider Example 1. Using our method described above, we can encode the constraints in Example 1 as the following formula.

\[
\neg U_{11} \lor \neg U_{21} \land \neg U_{12} \land \neg U_{22} \land \neg U_{13} \land \neg U_{23} \land \neg U_{14} \land \neg U_{24} \land \neg U_{15} \land \neg U_{25} \\
\land ((U_{11} \land U_{13}) \lor (U_{11} \land U_{14} \land U_{15})) \land ((U_{22} \land U_{23} \land U_{24}) \lor (U_{22} \land U_{25}))
\]

The above formula is satisfiable, thus there is a way to assign resources so that all the jobs can be accomplished. In fact, if we give the above formula to a Boolean satisfiability solver, the solver may return the following satisfying assignment.

\{U_{11}=true, U_{21}=false, U_{12}=false, U_{22}=true, U_{13}=true, U_{23}=false, U_{14}=false, U_{24}=false, U_{15}=false, U_{25}=true \}

The interpretation for the above satisfying assignment is simply that \( J_1 \) is assigned \( R_1 \) and \( R_3 \), \( J_2 \) is assigned \( R_2 \) and \( R_5 \).

Summary

In this paper, we consider a particular type of scheduling problem. We show that this problem can be reduced to the Boolean satisfiability problem. This reduction is based on a suitable encoding of the constraints in the scheduling problem. The encoding produces a Boolean formula, which has the property that all the jobs can be accomplished if and only if the Boolean formula is satisfiable and the satisfying assignment corresponds to possible scheduling. This method is scalable because the underlying Boolean satisfiability solvers are scalable and the reduction does not blowup the size of the problem.

Acknowledgement

This work is supported by Liaoning Provincial Natural Science Foundation under grant 201202202, Scientific Research Foundation of Liaoning Provincial Education Department under grant L2012388 and L2014440.

References


