Computers & Operations Research 73 (2016) 165–173

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Mixed Integer Programming models for job shop scheduling: A computational analysis

Wen-Yang Ku *, J. Christopher Beck

Department of Mechanical & Industrial Engineering, University of Toronto, 5 King's College Rd., Toronto, ON, Canada M5S 3G8

ARTICLE INFO

Article history:
Received 11 November 2014
Received in revised form
4 October 2015
Accepted 8 April 2016
Available online 9 April 2016

Keywords:
Job shop scheduling
Mixed Integer Programming
Constraint programming

ABSTRACT

In both industry and the research literature, Mixed Integer Programming (MIP) is often the default approach for solving scheduling problems. In this paper we present and evaluate four MIP formulations for the classical job shop scheduling problem (JSP). While MIP formulations for the JSP have existed since the 1960s, it appears that comprehensive computational studies have not been performed since then. Due to substantial improvements in MIP technology in recent years, it is of interest to compare the standard JSP models using modern optimization software. We perform a fully crossed empirical study of four MIP models using CPLEX, GUROBI and SCIP, focusing on both the number of instances that can be proved optimal and the solution quality over time. Our results demonstrate that modern MIP solvers are able to prove optimality for moderate-sized problems very quickly. Comparing the four MIP models, the disjunctive formulation proposed by Manne performs best on both performance measures. We also investigate the performance of MIP with multi-threading and parameter tuning using CPLEX. Noticeable performance gain is observed when compared to the results using only single thread and default parameter settings. Our results serve as a snapshot of the performance of modern MIP solvers for an important, well-studied scheduling problem. Finally, the results of MIP is compared to constraint programming (CP), another common approach for scheduling, and the best known complete algorithm to provide a broad view among different approaches.

1. Introduction

Mixed Integer Programming (MIP) has been widely applied to scheduling problems and it is often the initial approach to attack a new scheduling problem. For example, of the 40 research papers published in the *Journal of Scheduling* in 2014, 14 use MIP, more than any other technology. Given this popularity, together with the improvements in commercial MIP technology [1], it is valuable to understand how various MIP models for scheduling compare with each other in the context of modern solvers.

There are three widely used general MIP formulations for scheduling problems: the time-indexed formulation [2,3], the rank-based formulation [4], and the disjunctive formulation [5]. A theoretical comparison among these formulations claims that the disjunctive formulation as best and the time-indexed formulation worst [6]. However, despite the long history of these formulations, there does not appear to have been a complete computational study comparing the performance of different formulations using modern MIP solvers. In the most recent computational study we have found (from 1991), Applegate et al. [7] proposed an efficient branch and bound algorithm based on the disjunctive formulation and concluded that the JSP is computationally challenging even for moderate-sized problems.

In this paper, we compare the performance of four MIP models for the classical job shop scheduling problem (JSP). In addition to the three standard JSP formulations, we include a second disjunctive formulation claimed by Liao [8] to be superior to the original one. We perform experiments with three different solvers: IBM ILOG CPLEX v12.6.2 [9], GUROBI v6.0.4 [10], and SCIP v3.1.1 [11]. CPLEX and GUROBI are both regarded as the state-of-the-art commercial MIP solvers [1] while SCIP is the fastest non-commercial solver [12]. Our goal is to provide a complete empirical comparison of the MIP models for the JSP using different MIP solvers and identify the most efficient model.

Our experimental results using CPLEX and GUROBI show, contrary to Pan [6], that the time-indexed model is able to perform better than the rank-based model for small problems but fails to scale to larger problems due to the size of the model. In addition, contrary to Liao's finding, our results show that the original disjunctive model is more efficient than Liao's disjunctive model for both CPLEX and GUROBI. However, the experiments using SCIP provide the opposite conclusions: first, the rank-based model outperforms the time-indexed model for small problems and

* Corresponding author.

http://dx.doi.org/10.1016/j.cor.2016.04.006
0305-0548/

second, while at first glance Liao's disjunctive formulation seems more efficient than the original disjunctive formulation, careful investigation reveals that these differences are due to different preprocessing techniques undertaken by SCIP, and the phenomenon of erraticism [13] in the search process of MIP solvers. Despite these differences, across all tested solvers the disjunctive model outperforms the rank-based and the time-indexed models.

We then investigate two crucial aspects of modern MIP solvers: multi-threading and parameter tuning. Multi-threading allows the search to be executed in parallel and therefore may drastically improve performance. Parameter tuning aims at finding the best set of parameters for a specific class of problems. We perform experiments with the best MIP model, the disjunctive model, using CPLEX. Results show that running 8 threads in parallel improves the performance by about a factor of three. With multi-threading enabled, CPLEX's parameter tuning tool can further improve the performance by about a factor of 1.5 for problems that can be proved optimal, which is 4.5 times faster than the single-threaded default search. For the problems for which the optimal solution could not be found and proved within one hour, the solution quality over time is improved by about 30%.

We also provide a comparison of the best MIP results with two complete approaches: constraint programming (CP) [14,15] and iSTS-SGS [16], with the latter being the state-of-the-art exact algorithm. Our motivations of comparing CP and MIP are twofold. First, since MIP and CP are widely used by practitioners as "out-of-the-box" technology for scheduling, it is valuable to understand the differences between their performance. Second, due to the substantial improvements of MIP and CP solvers over the last decade [1], it is of interest to investigate the advances of MIP and CP, and how they compete with a state-of-the-art algorithm. We perform an empirical study using CPLEX for the disjunctive MIP model, with both multi-threading and parameter tuning and IBM ILOG CP Optimizer for the CP model. Our experimental results demonstrate that MIP performs similarly to CP for problems with moderate size and the goal of proving optimality. However, CP dominates MIP for the larger JSP instances and it is competitive with iSTS-SGS.

In summary, the contribution of this paper lies in the empirical analysis of the MIP models for the job shop scheduling problem. The rest of the paper is organized as follows. In Section 2 we give the definition of the JSP and review relevant literature. Section 3 describes the MIP models. Section 4 presents the computational results and the discussions. We conclude in Section 5.

2. Background

2.1. Problem definition

The JSP is defined by a finite set J of n jobs and a finite set M of m machines (Fig. 1). For convenience we refer an $n \times m$ problem. For each job $j \in J$, we are given a list $(\sigma_1^j, \ldots, \sigma_h^j, \ldots, \sigma_m^j)$ of the machines which represents the processing order of j through the machines. Note that σ_h^j is called the h-th *operation* of job j and σ_m^j is the last operation of job j. In addition, for each job j and machine i, we are given a non-negative integer p_{ij}, which represents the processing time of j on i. Each machine can process at most one job at a time, and once a job starts on a given machine, it must complete processing on that machine without interruption. The objective is to find a schedule of J on M that minimizes the makespan, i.e., the maximum completion time of the last operation of any job in J. Makespan minimization for the JSP is NP-hard for $n \geq 3$ and $m \geq 2$ [17].

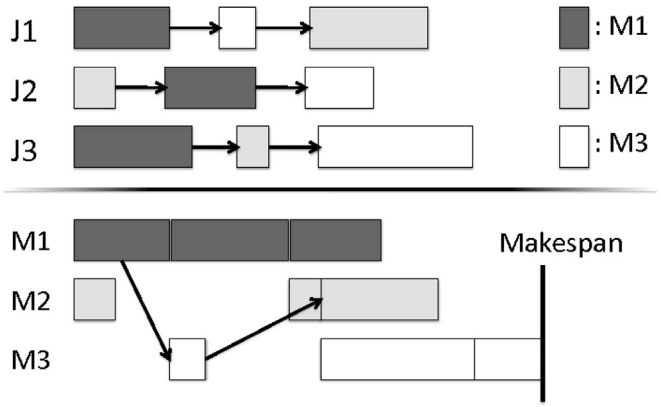

Fig. 1. Job shop scheduling problem. Three jobs J1, J2, and J3 are to be scheduled on three machines M1, M2 and M3. The graph on the top represents the precedence constraints. The Gantt Chart on the bottom displays a feasible schedule which satisfies the precedence constraints.

2.2. Literature review

As noted, the three standard MIP formulations for scheduling problems are the time-indexed formulation [2,3], the rank-based formulation [4], and the disjunctive formulation [5]. Other models are typically combinations or variations of these formulations. We define each model formally in the next section.

Pan [6] performed a theoretical comparison of these three models for the JSP, the flow shop scheduling problem, and the permutation flow shop scheduling problem. He concluded that the disjunctive model is most efficient since it has the fewest binary variables, followed by the rank-based model, and then the time-indexed model. While model size is an important aspect of the quality of a formulation, it is well-known that other characteristics such as the tightness of the linear relaxation can be equally or more important. Pan's evaluation of the models was challenged for the flow shop scheduling problem by Ronconi [18], who empirically compared the performance of two MIP models and showed that a model with half the number binary variables performed much worse that a larger model.

Liao [8] proposed a modified disjunctive formulation and showed empirically that this new model performs better than the original disjunctive model. The reduction in the computation effort was attributed to the fact that the new model had fewer linear constraints. While Liao showed that the computation time is significantly reduced with this new formulation, the experiments used problem instances with fewer than 5 jobs and 10 machines.

The literature on the JSP is vast with Google Scholar returning over 60,000 references (accessed September 28, 2015). Most approaches to optimization have been applied to JSP at some point, including heuristics [19], metaheuristics [20], genetic and evolutionary algorithms [21], customized branch-and-bound [7], constraint programming [22], and decomposition procedures [16].

3. MIP models

In this section, we present the four MIP models considered in this paper.

3.1. Disjunctive model

Our disjunctive model is based on Manne [5]. The decision variables are defined as follows:

- x_{ij} is the integer start time of job j on machine i