ABSTRACT

A heuristic algorithm that uses iteratively LPT and MF approaches on different job and machine sets constructed by using the current solution is developed to solve a classical multiprocessor scheduling problem with the objective of minimizing the makespan. Computational results indicate that the proposed algorithm is very competitive with respect to well-known constructive algorithms for a large number of benchmark instances.

Keywords: Different Job and Machine Sets, Empirical Results, Heuristic Algorithm, Identical Parallel Machines, Minimizing Makespan

INTRODUCTION

The scheduling problem of independent jobs on parallel machines is one of the most studied problem in combinatorial optimization both for its theoretical interest and for its practical aspect in many real world applications (see Lee, Lei & Pinedo, 1997). Still it continues to interest many researchers (Huang, Zhang, & Alexander, 2012; Montoya-Torres, Gómez-Vizcaino, Solano-Charris, & Paternina-Arboleda, 2010; Mungan, Yu, Sarker, & Rahman, 2012; Wang, Moraga, & Ghreyeb, 2011). In this paper we address the classical problem of scheduling a set $J=\{1,...,j,...,n\}$ of $n$ independent jobs, with positive processing times $p_j>0, j\in J$ and simultaneously available, on a set $M=\{1,...,i,...,m\}$ of $m$ identical parallel machines. Each machine can process at the most one job at a time, and each job must be processed without interruption by exactly one of the $m$ machines. The paper considers the problem of finding the schedule that minimizes the maximum job completion time (makespan). This problem is denoted in the literature as $P||C_{max}$, see Graham, Lawler, Lenstra, & Rinnooy Kan, 1979.

A feasible solution (schedule) for $P||C_{max}$ is represented by an $m$-partition $S=\{S_1,...,S_s,...,S_m\}$ of the set $J$, where each $S_i$ represents the subset of jobs assigned to the machine $i$, $i\in M$. For each feasible solution $S$, the work-loads of the machines are represented by the $m$-set $C(S)=\{C(S_1),...,C(S_s),...,C(S_m)\}$, where $C(S_i)=\sum_{j\in S_i}p_j$ is the work-load of machine $i\in M$. In other

DOI: 10.4018/ijoris.2015010102
words, \( C(S) \) represents the completion time of the latest job assigned to the machine \( i \in M \). For each schedule \( S \), \( C_{max}(S) = \max_{i \in M} \{ C(S) \} \) denotes the makespan associated with the solution \( S \).

\( P||C_{max} \) is strongly NP-Hard for an arbitrary \( m \geq 2 \) (see Garey & Johnson, 1979). It is supposed that \( n \geq m \geq 2 \) to avoid trivialities. Exact algorithms have been proposed in Blazewicz, 1987; Dell’Amico & Martello, 1995; Mokotoff, 2004; Dell’Amico, Iori, Martello, & Monaci, 2008. However for large instances it needs to design good heuristic algorithms to obtain, in reasonable time, solutions that are probably close to the optimum. Improvement heuristics have been proposed in Finn & Horowitz, 1979; Langston, 1982; Fatemi Ghomi & Jolai Ghazvini, 1998. In addition, metaheuristic procedures have been developed in Hübscher & Glover, 1994; Thesen, 1998; Dell’Amico et al., 2008; Paletta & Vocaturo, 2011.

Many polynomial time constructive approaches, having a known worst-case performance ratio, have been developed to solve \( P||C_{max} \) (for an overview, see Cheng & Sin, 1990; Lawler, Lenstra, Rinnooy Kan, & Shmoys, 1993; Hoogeveen, Lenstra, & Van de Velde, 1997; Chen, Potts, & Woeginger, 1998; Mokotoff, 2001). The most important constructive approaches to solve \( P||C_{max} \) are:

- **List Scheduling family (LS) (Graham, 1966; 1969):** A list of all the jobs is made and then iteratively the first job is taken out of the list and assigned to the least loaded machine. The process continues until the list is empty. There are different kinds of LS algorithms, depending on how the list of jobs is made. The best known LS algorithm is the Longest Processing Time (LPT) that first sorts all jobs into a list in non-increasing order with respect to their processing times, and then it iteratively assigns the uppermost job from the list to the least loaded machine, until the list is exhausted. The LPT algorithm runs in \( O(n \log n) \)-time, where the first term corresponds to ordering jobs, and the second to the assignment of each job to a machine. Its worst-case ratio is equal to \( 4/3 - 1/(3m) \), which is tight;

- **MultiFit (MF) (Coffman, Garey, & Johnson, 1978):** A number of times a Bin Packing Problem (BPP) with an appropriate capacity is iteratively solved. BPP can be seen as a “dual” of \( P||C_{max} \) in which the jobs must be assigned to an unlimited number of identical machines, without exceeding a prefixed capacity (makespan) \( c \), with the objective to minimize the number of machines used. The idea behind MF is to find (by binary search) the smallest prefixed machine capacity \( c \) such that no more than \( m \) machines need to accommodate all jobs (that are taken in non-increasing order of \( p_j \) and each job is placed into the first machine into which it will fit. Coffman et al., 1978 show that MF runs in \( O(n \log n + t n \log m) \)-time and its worst-case ratio is equal to \( 1.22 + 2^{-t} \), where \( t \) represents the number of times that a BPP is solved. It is recommended that \( t \) be at least 7. Friesen, 1984, improves this bound from 1.22 to 1.2, and Yue, 1990, improves it to 13/11, which is tight.

Lee & Massey, 1988, subsequently propose the COMBINE heuristic which utilizes the solution of LPT as the incumbent and then applies MF with fewer iterations. COMBINE requires the same computational time and has the same worst-case ratio as MF. Gupta & Ruiz-Torres, 2001, propose the LISTFIT algorithm that is based on the combination of multiple list of jobs with the bin packing approach. LISTFIT runs in \( O(n \log n + t n^2 \log m) \)-time and its worst-case ratio performance is like MF. Other important constructive methods are the Largest Differentiating Method (LDM) (see Karmarkar & Karp, 1982), and Multi-Subset (see Dell’Amico & Martello, 1995). A more recent algorithm, that is based on the idea of combining iteratively partial solutions, is described in Paletta & Pietramala, 2007, and Paletta & Vocaturo, 2010.

In this paper, an \( O(n \log n + m t n \log m) \) algorithm, called Different Job and Machine Sets (DJMS), is designed to solve \( P||C_{max} \). It
Related Content

Sourcing Strategies and Theories
(2013). Knowledge Driven Service Innovation and Management: IT Strategies for Business Alignment and Value Creation (pp. 325-368).
www.igi-global.com/chapter/sourcing-strategies-theories/72482?camid=4v1a

A Step-by-Step Procedural Methodology for Improving an Organization’s IT Risk Management System
www.igi-global.com/chapter/a-step-by-step-procedural-methodology-for-improving-an-organizations-it-risk-management-system/202221?camid=4v1a

Preconditions for the Management of Invention-Innovation Diffusion Process
www.igi-global.com/article/preconditions-for-the-management-of-invention-innovation-diffusion-process/101330?camid=4v1a
Green Production and Green Supply Chain Governance Under Extended Producer Responsibility
Xiukun Zhao, Xuanming Bai, Xiaobing Lv and Yongjian Li (2018). Green Production Strategies for Sustainability (pp. 1-26).
www.igi-global.com/chapter/green-production-and-green-supply-chain-governance-under-extended-producer-responsibility/192825?camid=4v1a