Theoretical Computer Science 555 (2014) 9–22

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

The cyclical scheduling problem

Binay Bhattacharya [a], Soudipta Chakraborty [b], Ehsan Iranmanesh [a], Ramesh Krishnamurti [a,*]

[a] *Simon Fraser University, Canada*
[b] *Indian Institute of Technology, Kharagpur, India*

ARTICLE INFO

Article history:
Received 7 June 2013
Received in revised form 21 February 2014
Accepted 25 March 2014
Available online 2 April 2014

Keywords:
Weighted matching
b-Matching
Cyclic scheduling
Balanced binary tree
Network flow

ABSTRACT

We consider the $(n-2, n)$ cyclical scheduling problem which assigns a shift of $n-2$ consecutive periods among a total of n periods to workers. We solve this problem by solving a series of b-matching problems on a cycle of n vertices. Each vertex has a capacity, and edges have costs associated with them. The objective is to maximize the total cost of the matching. The best known algorithm for this problem uses network flow, which runs in $O(n^2 \log n)$ on a cycle. We provide an $O(n \log n)$ algorithm for this problem. Using this, we provide an $O(n \log n \log nb_{max})$ algorithm for the $(n-2, n)$ cyclical scheduling problem, where b_{max} is the maximum capacity on a vertex.

1. Introduction

The cyclical scheduling problem is used to schedule shifts for workers in a factory. Given a set of n work periods, each worker is assigned a shift where he works for $n-2$ consecutive periods, and takes off the remaining 2 periods. Thus, for $n=7$, a typical shift may be to work from Monday to Friday and take off Saturday and Sunday. Another shift may be to work from Friday to Tuesday and take off Wednesday and Thursday (there are 7 such shifts in a week). Each shift may also have a cost associated with it. In addition, the factory requires that a given number of workers be available each period (this requirement may vary from period to period). The objective is to assign a shift to each worker such that the daily requirement is fulfilled and the total cost of the shifts is minimized. We solve the cyclical scheduling problem by solving a series of b-matching problems.

Tibrewala et al. [19] provide an integer programming formulation for the problem when all shift costs are equal. They also provide a simple algorithm to solve the problem optimally. Bartholdi and Ratliff [4] solve the cyclical scheduling problem by considering the "complementary problem" where given the number of workers w in the factory, they maximize the number of workers who are off for two consecutive periods. There is an upper bound on the number of workers who are off during each period. The number of workers they start out with, w, is adequate if the objective function value is at least as much as w. A binary search procedure is used to find the minimum value for w such that w is adequate.

Bartholdi et al. [5] extend this approach to the general (k, n) cyclic scheduling problem, where there is a total of n periods, and each worker is assigned a shift of k consecutive periods to work (and is off for the remaining $(n-k)$ periods). There is a cost associated with each shift, and the objective is to minimize the total cost of shifts assigned to workers. They

* Corresponding author.
E-mail addresses: binay@sfu.ca (B. Bhattacharya), soudipta.c@gmail.com (S. Chakraborty), iranmanesh@sfu.ca (E. Iranmanesh), ramesh@sfu.ca (R. Krishnamurti).

http://dx.doi.org/10.1016/j.tcs.2014.03.034
0304-3975/

provide a parametric solution to this general problem by solving a series of network flow problems, one for each guess for the workforce size. In addition, they also provide a solution using linear programming and roundoff. Using the above notation, we address the $(n-2, n)$ cyclic scheduling problem in this paper. Alfares [1] provides computational results using linear programming for the $(5, 7)$ cyclic scheduling problem. Alfares [2] proposes an integer programming formulation for the cyclic scheduling problem where each worker gets 3 days off each week, with additional constraints, such as having at least 2 of the 3 off days consecutive. In a more recent paper, Hochbaum and Levin [13] provide a quadratic algorithm when all shift costs are equal. Hochbaum and Levin [12] generalize the cyclical scheduling problem where each worker is assigned multiple shifts, where each shift is a sequence of consecutive work periods followed by a sequence of off periods. They show that this problem is NP-hard when there are two or more such shifts, and propose approximation algorithms for the problem.

The very first polynomial time algorithm for finding the maximum size b-matching was proposed by Johnson [14]. It was an extension of Edmond's algorithm [9]. Gabow [10] also provided an $O(nm \log n)$ time algorithm for finding the maximum size b-matching (n is the number of vertices and m is the number of edges). The b-matching polytope (convex hull) of a graph was characterized by Edmonds and Pulleyblank [18]. Pulleyblank [17] has also provided a pseudo-polynomial algorithm to find a maximum weighted b-matching. The first polynomial time algorithm to solve optimal maximum weighted b-matching was given by Cunningham and Marsh [7]. Marsh [15] also extended the Edmond weighted matching algorithm for finding the maximum weighted b-matching. Anstee [3] and Gerard [11] found strongly polynomial algorithms for this problem.

2. The b-matching formulation

We provide the formulation for the cyclical scheduling problem with n periods, with each shift comprising $n-2$ consecutive work periods and 2 off periods. Shift s_i assigns off periods i and $(i+1) \bmod n$ and the remaining as working periods. x_i denotes the number of workers assigned to shift s_i and c_i denotes its cost. b_i denotes the number of workers required to work during period i. The total cost of the assignment is given by $\sum_{i=0}^{n-1} c_i x_i$. The left hand side of each constraint is the number of workers working on shift s_i, and is given by $x_{(i+2) \bmod n} + x_{(i+3) \bmod n} + \cdots + x_{(i-1) \bmod n}$. This should be no less than the requirement b_i. Without loss of generality, we assume all index arithmetic is done modulo n in the formulations for Problem I and Problem II that follow:

$$\textbf{Problem I} \quad \left[\begin{array}{lll} \text{Minimize} & S = \sum_{i=0}^{n-1} c_i x_i & \\ \text{s.t.} & x_{i+2} + x_{i+3} + \cdots + x_{i-1} \geqslant b_i & \forall i,\ 0 \leqslant i \leqslant n-1 \\ & x_i \geqslant 0 \text{ and integer} & \forall i,\ 0 \leqslant i \leqslant n-1 \end{array}\right.$$

The above can be reformulated as a b-matching problem. If $w = \sum_{i=0}^{n-1} x_i$ denotes the total number of workers, the number of workers working on shift s_i during period i is given by $w - x_{i-1} - x_i$. Letting $d_i = w - b_i$, each of the above inequalities can be rewritten as $x_{i-1} + x_i \leqslant d_i$, where d_i denotes the number of workers that are off during period i. We thus obtain the formulation for Problem II below, given by Bartholdi and Ratliff [4]:

$$\textbf{Problem II} \quad \left[\begin{array}{lll} \text{Maximize} & S = \sum_{i=0}^{n-1} c_i x_i & \\ \text{s.t.} & x_{i-1} + x_i \leqslant d_i & \forall i,\ 0 \leqslant i \leqslant n-1 \\ & x_i \geqslant 0 \text{ and integer} & \forall i,\ 0 \leqslant i \leqslant n-1 \end{array}\right.$$

The term $c_i x_i$ in the objective function corresponds to the cost that is saved in period i because x_i workers in shift s_i are off during this period. The objective function thus maximizes the total cost saved over the n periods. The above formulation for Problem II can be visualized as a graph $G = (V, E)$ with vertex set $V = \{v_0, v_1, \ldots, v_{n-1}\}$ and edge set $E = \{e_0, e_1, \ldots, e_{n-1}\}$, where $e_0 = (v_0, v_1)$, $e_1 = (v_1, v_2), \ldots, e_{n-1} = (v_{n-1}, v_0)$. We associate capacities $d_0, d_1, \ldots, d_{n-1}$ with the vertices $v_0, v_1, \ldots, v_{n-1}$ respectively.

We describe in this paper an algorithm based on augmenting paths to solve the above problem. We start with an estimate for w, the total number of workers. If $\bar{w} = \sum_{i=0}^{n-1} x_i < w$, where x_i for each i is the assignment obtained from the optimal solution to Problem II, and w is our estimate of the number of workers, then we revise this estimate upward. As observed by Bartholdi and Ratliff [4], a binary search may be performed to find the smallest value of w for which $\bar{w} \geqslant w$. By solving Problem II for this value of w, we obtain the optimal assignment of shifts to workers.

If w^* is the optimal value of w returned by the binary search procedure, then $b_{max} \leqslant w^* \leqslant \sum_0^{n-1} b_i$, where $b_{max} = \max_i\{b_i\}$. Bartholdi and Ratliff [4] also proved the following inequalities for a particular value $w = w'$ and its corresponding solution $\{x_i\}$:

$$\sum_{i=0}^{n-1} x_i < w' \quad \text{iff} \quad w' < w^* \tag{1}$$