Available online at www.sciencedirect.com
ScienceDirect
Discrete Optimization 3 (2006) 327–340
DISCRETE OPTIMIZATION
www.elsevier.com/locate/disopt

Cyclical scheduling and multi-shift scheduling: Complexity and approximation algorithms

Dorit S. Hochbaum[a], Asaf Levin[b,*]

[a] *Department of Industrial Engineering and Operations Research and Walter A. Haas School of Business, University of California, Berkeley, United States*
[b] *Department of Statistics, The Hebrew University, Jerusalem, Israel*

Received 1 March 2004; received in revised form 4 February 2006; accepted 13 February 2006
Available online 13 July 2006

Abstract

We consider the multiple shift scheduling problem modelled as a covering problem. Such problems are characterized by a constraint matrix that has, in every column, blocks of consecutive 1s, each corresponding to a shift. We focus on three types of shift scheduling problems classified according to the column structure in the constraint matrix: columns of consecutive 1s, columns of cyclical 1s, and columns of k consecutive blocks. In particular, the complexity of the cyclical scheduling problem, where the matrix satisfies the property of cyclical 1s in each column, was noted recently by Hochbaum and Tucker to be open. They further showed that the unit demand case is polynomially solvable. Here we extend this result to the uniform requirements case, and provide a 2-approximation algorithm for the non-uniform case. We also establish that the cyclical scheduling problem's complexity is equivalent to that of the exact matching problem—a problem the complexity status of which is known to be randomized polynomial (RP). We then investigate the three types of shift scheduling problems and show that, while the consecutive ones version is polynomial and the k-block columns version is NP-hard (for $k \geq 2$), for the k-blocks problem we give a simple k-approximation algorithm, which is the first approximation algorithm determined for the problem.

Keywords: Multiple shift scheduling problem; Cyclical scheduling

1. Introduction

In labor scheduling problems there are requirements for a specific number of workers at each time period. The problem is to assign workers to a collection of time periods consisting of work schedules so that the requirements are met with the least cost or least total number of workers. This generic description of the problem is formulated as the *multicover* problem. There are n possible schedules, each included x_j times, and m time periods, each with the requirement of b_i workers. The constraint coefficient matrix is (a_{ij}) with $a_{ij} = 1$ if period i is covered by schedule j.

* Corresponding address: Israel Institute of Technology, Industrial Engineering and Management, Technion, Haifa, Israel.
E-mail addresses: hochbaum@ieor.berkeley.edu (D.S. Hochbaum), levinas@mscc.huji.ac.il (A. Levin).

1572-5286/$ - see front matter
doi:10.1016/j.disopt.2006.02.002

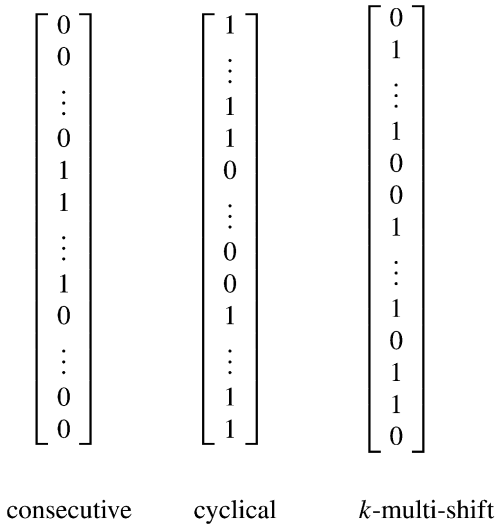

Fig. 1. Three types of columns of MC.

The formulation as multicover (MC) is,

$$\begin{aligned} &\text{(MC)} \quad \text{Min} \quad \sum_{j=1}^{n} c_j x_j \\ &\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, \ldots, m \\ &\qquad\qquad x_j \geq 0 \text{ integer}, \quad j = 1, \ldots, n. \end{aligned}$$

Constraint i represents the requirement that at least b_i workers' schedules have to be selected. A column of the constraint matrix represents one possible work schedule, also known as *rotation*. Each rotation represents a collection of work periods that can be assigned to a single worker, or a single crew, within the planning period. The cost c_j is 1 if the goal is to minimize the total number of workers, or it is the total cost for the specific schedule that can reflect overtime pay and other rotation-related expenses such as lodging and meals for airline crews.

In case there is a limit on the number of workers for each type of work schedule, then we have an upper bound u_j on the number of workers that can be assigned to work schedule of type j. The non-negativity constraints are then replaced by,

$$0 \leq x_j \leq u_j \text{ integer}, \quad j = 1, \ldots, n$$

and we refer to the problem as MCB (MultiCover Bounded). When all requirements $b_i = 1$, the problem is the well-known *set cover* problem.

The family of covering problems that include MC and MCB is notoriously hard among intractable and NP-hard problems. These problems are commonly used in modelling optimization problems in many diverse applications. There is hence an extensive literature on integer programming tools to address the problems and on heuristics and approximation algorithms. Here we investigate the structure of the multi-shift scheduling problem towards the end of identifying more efficient procedures than are possible for the general MC and MCB problems.

The particular structure of shift scheduling problems becomes apparent when the rows are ordered in correspondence to the ordering of the time line. The formulation of the multicover problem representing shift scheduling tends to then have columns of the constraint matrix that form intervals of consecutive ones. This is because work shifts have consecutive time periods as a matter of convenience and productivity. The simplest type of constraint matrix has columns of *consecutive* ones, such as those displayed in Fig. 1. This corresponds to rotations consisting of single shifts.

We call a zero/one column vector a *consecutive ones vector*, a *circular ones vector* and a *k-multi-shift vector* if it has at most one block of ones, at most one block of zeros, and at most k block of ones, respectively (see Fig. 1 for

an illustration of these definitions). Obviously, every consecutive 1s matrix is also cyclical, and every cyclical matrix is also 2-multi-shift. Any general multicover matrix can be considered a k-multi-shift matrix, where the number of blocks can be as large as the number of 1s in a column. We focus here on k-multi-shift matrices with a small value of k.

A matrix with consecutive 1s in each column (or row) is totally unimodular. This was shown by Veinott and Wagner [23] who devised a unimodular transformation that transforms the constraint matrix into a matrix with at most one 1 and one -1 in each column. Since matrices with one 1 and one -1 in each column are node-arc incidence matrices, which are known to be totally unimodular, the respective optimization problem can be solved with network flow techniques. The transformation is shown in Section 2.

Planning problems *cyclical scheduling* and *k-multi-shift scheduling* are formulated as MC problems with a cyclical matrix and k-multi-shift matrix, respectively. The k-multi-shift scheduling problem corresponds to an application where each rotation is composed of at most k *busy periods*. A rotation typically consists of a number of consecutive time periods of work separated by consecutive time periods of break, and then again consecutive time periods of work etc. When the time periods are hours, each set of consecutive hours forms a shift. In scheduling police, hospital or fire department workers, the rotations more typically form a series of consecutive sequences of days separated by intermittent breaks of several days off. The time horizon then is usually a month or a quarter. A similar structure is apparent also for airline crew scheduling.

1.1. Related research

The question of shift scheduling has been addressed extensively in the operations research literature since the 1960s. A great deal of research focused on lower bounds on the least number of workers needed, on integer programming techniques [17], and on generating feasible solutions [15]. The majority of the work, with the exception of studies of schedules relevant to the airlines, has focused on the unweighted problem where the objective is to minimize the number of workers, Min $\sum_{j=1}^{n} x_j$.

There have been several research papers attempting to generate solutions to the k-multi-shift scheduling problem that are close to the optimum in terms of absolute error, as in the work of Bartholdi [5], Morris and Showalter [19] and Vohra [24]. The reported bounded error algorithms in [5,19] rely on a solution to a linear programming relaxation of the multicover problem, and Vohra [24] showed, for a special subclass of problems, a bound similar to that of Bartholdi's [5] that can be generated with a linear time algorithm. These bounds are all on the absolute error, which is a value Δ so that, for a problem with an optimal solution value OPT and a heuristic solution value H, $|H - OPT| \leq \Delta$. To the best of our knowledge, no approximation algorithms have been analyzed for the problem, which would, for a δ-*approximation algorithm*, have a guaranteed upper bound on the *relative* error, $\frac{H}{OPT} \leq \delta$. Nor has there been any form of error analysis that depends on the number of blocks per column, which translates to meaningful information in terms of the type of feasible work schedules. The algorithm of Morris and Showalter [19] has an absolute error bound of m for the unweighted MC problem. The algorithms of Bartholdi [5] and of Vohra [24] both deliver an absolute error bound of $(q-1)\frac{n}{k}$, where q is the maximum number of blocks *per row* of the constraint matrix, and k is the number of 1s per column, which is restricted to be the same for all columns.

Bartholdi et al. [4] proposed a polynomial time algorithm for the cyclical scheduling problem that also has the property of circular 1s (in rows). They noted that the column cyclicality reflects the cyclic nature of the work shifts, but it is the row circularity that was used to obtain efficient solutions. Cyclical scheduling problems that are also circular arise in (k, n) cyclical scheduling problems where the feasible work schedules and columns consist of all possible k consecutive 1s in a column of length n. For such problems, the row structure of the constraint matrix is circular as well.

As mentioned earlier, Veinott and Wagner [23] studied the covering problem with consecutive 1s in columns and established its relationship to the minimum cost network flow problem—a relationship which we describe in detail in Section 2.

Adamy et al. [1] considered the "call control" problem. The objective of this problem is Max $\sum_{j=1}^{n} x_j$ subject to packing constraints $Ax \leq b$ and $x \in \{0, 1\}$, where A is a cyclical matrix. They presented a linear time algorithm that solves the "call control" problem.

Erlebach [8] observed that the "call control" algorithm can be used to solve a special case of the cyclical unweighted MCB, where the upper bounds u_j are polynomially bounded in the dimensions of A (i.e., in m and n). Given an

Table 1
Summary of algorithms and complexity for multi-shift scheduling problems

Problem	Complexity	Reference
Consecutive	Same as minimum cost network flow	[23]
Cyclical with all $b_i = 1$	$O(m(n + m \log n))$	[12]
Cyclical with all $b_i = B$	$O(n^3 \log B + n^4)$	Here
Cyclical with arbitrary r.h.s.	Equivalent to exact matching (likely polynomial)	Here
k-multi-shift matrices	NP-hard even for $k = 2$; k-approximation	Here

unweighted MCB problem, replace each variable x_j by a sum of u_j binary variables. The result is an instance of MCB where each upper bound is 1 with constraint matrix A'. Substituting $y_j = 1 - x_j$ for all j results in a packing problem, Min $\sum_{j=1}^{n}(1 - y_j)$ subject to $A'y \leq b'$ and $0 \leq y \leq 1$, where $b'_i = \sum_j A'_{ij} - b_i$. The objective of this problem is equivalent to Max $\sum_{j=1}^{n} y_j$. This latter packing problem is the "call control" problem, and one can use the linear time algorithm of Adamy et al. [1] to solve the problem in polynomial time. The time complexity of the resulting algorithm for the cyclical unweighted MCB is $O(\sum_{j=1}^{n} u_j)$. Therefore, Erlebach's transformation [8] forms a matrix A' that has polynomial size if and only if the upper bounds u_j are polynomially bounded in m and n. This restriction is not satisfied in many cases, as there is no clear connection between the maximum right-hand side and the dimensions of the matrix.

Hochbaum and Tucker [12] considered the "bitonic minimax" problem. They showed that the bitonic minimax problem is equivalent to the set cover problem with a cyclical 1s matrix. They solved this problem in strongly polynomial time which, for a problem on n variables and m constraints, is $O(m(n + m \log n))$. They also showed that, for meaningful cyclical matrices, n is larger than m, and n is at most $O(m^2)$. Atallah et al. [2] devised an $O(n)$ time algorithm for computing the single-source shortest paths in a weighted circular-arc graph, where n is the number of circular-arcs. Using this algorithm, the complexity of Hochbaum and Tucker's [12] algorithm for the set cover problem with a cyclical 1s matrix improves to $O(mn)$. Moreover, Atallah et al. [2] used their single-source shortest paths algorithm for a circular-arc graph to present an $O(qn)$ time algorithm for the set cover problem with a cyclical 1s matrix, where q is the minimum number of intervals that cover a node of the cycle (i.e., the minimum row sum). We note that, in the worst case, $O(qn) = O(n^2)$, which could be worse than $O(mn)$.

The general set cover and general multicover are "hard" to approximate better than $\ln m$, as was demonstrated by Lund and Yannakakis [18]. Chvátal [6] extended an earlier greedy algorithm, devised independently by Johnson and Lovász, that applied to the unweighted set cover, to the weighted set cover with a worst-case bound of $\mathcal{H}(d)$, where $\mathcal{H}(d) = \sum_{i=1}^{d} \frac{1}{i}$ and d is the size of the largest set (or column sum). $\mathcal{H}(d)$ is bounded by $1 + \log d$. Dobson [7], extended this approximation algorithm to the multicover problem. Hochbaum [13] devised a δ-approximation algorithm for the (weighted) set cover where δ is bounded by the maximum row sum, and Hall and Hochbaum [10] extended the algorithm to apply, with the same bound, to the multicover problem as well.

1.2. Our results

To provide a context to the contributions here, we summarize the status of k-multi-shift problems in increasing difficulty in Table 1:

The transformation of Veinott and Wagner is described in Section 2. This transformation is critical to our results for the cyclical scheduling problem as well. In Section 3, we first show that the k-shift problem is NP-hard for all constant values of k with $k \geq 2$. This behaviour is different from the multicover problem that is polynomially solvable if each column has at most two 1s. Then we present a k-approximation algorithm for the k multi-shift problem. We conclude this section with a description of the linking of the complexity status of the cyclical scheduling problem to the open status of the exact matching problem. Finally, Section 4 describes the polynomial time algorithm for cyclical scheduling with the uniform requirements case in the uncapacitated case. Here, uniform requirements mean that all right-hand sides are identical, that is, $b_1 = b_2 = \cdots = b_m$.

2. Transformation of matrices with consecutive 1s

Veinott and Wagner [23] proposed the following transformation for a matrix with consecutive 1s in all columns. The transformation is to replace the ith row $A_{i\cdot}$ by $A_{i\cdot} - A_{i+1\cdot}$ for $i = 1, \ldots, m-1$. In other words, we subtract from

each row the next row. This transformation is equivalent to premultiplying the $m \times n$ matrix A by the unimodular $m \times m$ matrix T, where T has at most one 1 and one -1 per row:

$$T = \begin{bmatrix} 1 & -1 & & \dots & 0 & 0 & 0 \\ 0 & 1 & -1 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & \dots & 0 \\ \vdots & & & & & & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix}.$$

Moreover, if $Tx = y$, then $x_i = \sum_{j=i}^{m} y_j$ for $i = 1, \dots, m$. For $p \le q$, we denote the jth column $A_{\cdot j}$ of A, with $a_{pj} = \cdots = a_{qj} = 1$ and other entries equal to 0, by $[p, q]$. The consecutive 1s column $[p, q]$ is transformed to a column $TA_{\cdot j}$ with $a_{p-1,j} = -1$ (where $p - 1$ could be 0) and $a_{qj} = 1$:

$$\begin{matrix} \\ p \\ \\ \\ q \end{matrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \longrightarrow \begin{matrix} p-1 \\ p \\ \\ \\ q \end{matrix} \begin{bmatrix} 0 \\ \vdots \\ -1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

For $p > q$, the notation $[p, q]$ denotes a cyclical 1s column of the form $(1, \dots, 1, 0, \dots, 0, 1, \dots, 1)$, where $a_{q+1,j} = \cdots = a_{p-1,j} = 0$ and all other values are 1. That is, the indexing of the rows is modulo m. When applying the transformation T to a cyclical 1s column $[p, q]$ we get,

$$\begin{matrix} q+1 \\ \\ \\ \\ \\ p-1 \end{matrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \longrightarrow \begin{matrix} q \\ q+1 \\ \\ \\ \\ p-1 \end{matrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

The transformation can be applied only to a system of equalities. Since the problem MC is a covering problem $Ax \ge b$ (all inequalities are $\ge$ inequalities), we subtract surplus variables and get an equivalent system of equations $[A, -I] \cdot [x, s] = b$, where I is the identity matrix, s is the vector of surplus variables, and $s \ge 0$. Each column of $T \cdot (-I)$ is of the form $(0, \dots, 0, 1, -1, 0, \dots, 0)$.

It is thus apparent that a covering problem on a cyclical 1s constraint matrix is equivalent to a network flow problem with one "complicating" constraint—the one corresponding to the last row of the matrix. To see that, consider the formulation of a minimum cost network flow problem on a graph $G = (V, A)$, with y_{ij} being the variable indicating the amount of flow on arc (i, j) (one variable per each column), the cost of the flow on that arc, C_{ij}, is the cost of the variable x_k whose column corresponds to (i, j), and the capacity upper bound on that arc, U_{ij}, is again the upper

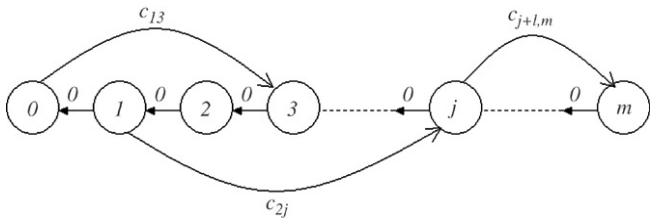

Fig. 2. The network that corresponds to a consecutive 1s matrix A.

bound u_k on the variable x_k defined as before. The supply/demand of node i is $d_i = b_i - b_{i+1}$[1]:

$$\begin{aligned} \text{Min} \quad & \sum_{(i,j)\in A} C_{ij} y_{ij} \\ \text{subject to} \quad & \sum_{k:(k,i)\in A} y_{ki} - \sum_{j:(i,j)\in A} y_{ij} = d_i, \quad i \in V \\ & 0 \le y_{ij} \le U_{ij}, \qquad\qquad (i,j) \in A. \end{aligned}$$

The complicating constraint is of the form $\sum_{(i,j)\in A'} y_{ij} \ge B$, where $A' \subseteq A$ is a subset of the columns (variables) that have an entry 1 in the last row. Note that, after applying the transformation T, the surplus variables in the last row of $T\cdot(-I)$ remain unchanged, and thus the constraint remains as it was before applying the transformation. A' includes all "strictly" cyclical 1s columns that have an entry 1 in both the first and last rows (before the transformation). After the transformation, these columns have the value 1 in the last row in addition to one -1 and one 1. The arcs in the network flow problem are generated as follows. For $p < q$ the column $[p, q]$ maps to the arc $(p-1, q)$. The ith column of $T\cdot(-I)$ maps to the arc $(i+1, i)$ (see Fig. 2). For $p > q$ the cyclical column $[p, q]$ maps to the *restricted* arc $(p-1, q)$ in A'.

Although the multicover problem on a cyclical 1s matrix can be written as a network flow problem where the sum of flows on a subset of the arcs is bounded, the reverse is not necessarily the case. For the given ordering of the rows of the matrix, there are three types of columns and corresponding arcs in the subgraph of the network that arise in the matrix $T\cdot[A, -I]$. For consecutive 1s columns an acyclic graph corresponds to the topological ordering implied by the ordering of the rows. That is, each arc goes from a lower index node to a higher index node, so for an arc (i, j), $i < j$. In terms of the column, this rule is reflected by a -1 appearing above the 1 entry in the column. The second type of subgraph arcs corresponds to cyclical columns, consisting of arcs that are directed opposite to the topological ordering, so for an arc (i, j) that corresponds to such column, $j < i$. The third type of arc corresponds to the surplus variables where the 1 appears in the one row above the -1, which corresponds to an arc $(i+1, i)$. The lower bound on the sum of flows applies then only for the cyclical column arcs that are restricted.

3. Complexity

From the transformation described in Section 2, it follows that the multicover problem on consecutive 1s matrices is polynomially solvable: apply T, then solve the resulting minimum cost network flow problem, and apply the inverse transformation T^{-1}. To see the complexity of the MC or MCB problem for the consecutive 1s matrices, we note that the transformed problem is an uncapacitated minimum cost network flow for MC and capacitated for MCB (see [23]).

The run time of the algorithm of Orlin [21] for this minimum cost network flow problem on a network $G = (V, A)$ is $O(|V|\log|V|(|A|+|V|\log|V|))$ and $O(|A|\log|V|(|A|+|V|\log|V|))$ for the uncapacitated and capacitated cases, respectively. In our case, $|A| = m + n$ and $|V| = m$. Therefore the resulting run times are $O(m\log m(n + m\log m))$ and $O(n\log m(n + m\log m))$ for MCB and MC, respectively.

3.1. The complexity of k-multi-shift problem

For $k \ge 3$, the complexity of the k-multi-shift problem is straightforward, since it is a multicover problem and the multicover problem with at least three 1s per column is NP-hard.

[1] We slightly abuse the standard notation of network flow problems where the flow balance constraint is written as the outflow from node i minus the inflow into node i. Our constraint is written as inflow minus outflow, which is equivalent to reversing the directions of all arcs in the network.

We now argue that the multicover problem where each column has at most two 1s is polynomially solvable. The constraint matrix can be interpreted as the edge-node incidence matrix of a graph $G = (V, E)$. In G we look for a variant of an edge cover of minimum cost in which each node v has to be covered b_v times (where b_v is some constant). This last problem can be transformed into a weighted b-matching problem (similarly to the transformation of the usual edge cover problem into minimum cost perfect matching). The claim holds since the weighted b-matching problem can be solved in polynomial time (see [22] pages 554–556).

This means that it is not clear whether the 2-multi-shift problem is polynomially solvable. We next show that it is NP-hard.

Theorem 1. *The 2-multi-shift problem is NP-hard.*

Proof. The SAT problem is defined on a set U of variables, a collection C of disjunctive clauses of literals, where a literal is a variable or a negated variable in U. The problem is to decide whether there is a truth assignment for U that satisfies all clauses. It is known that the SAT problem is NP- complete even when restricted to instances such that each variable appears exactly three times (see problem LO1 in [9]). The restriction of the SAT problem to such instances will be denoted as R3-SAT (standing for Restricted to 3). We denote $U = \{Y_1, Y_2, \ldots, Y_n\}$, and assume w.l.o.g. that each variable appears negated exactly once or exactly twice.

We now present a reduction from the R3-SAT instance denoted as I to an instance I' of the 2-multi-shift problem. For every appearance in I of Y_i in $j \in C$, I' has a variable X_{ij} corresponding to Y_i (note that there are at most three values of j). Next, we define the set of constraints of I'. The first block of constraints corresponds to constraints that allows us to either select a true assignment to a variable in I or a false assignment. These sets of constraints will have a pair of constraints for each Y_i. Denote by X_{i1}, X_{i2} and X_{i3} the three variables in I' corresponding to the three appearances of Y_i in I. If X_{i1} corresponds to Y_i and X_{i2}, X_{i3} correspond to $\overline{Y}_i$, then we have the following pair of consecutive constraints $X_{i1} + X_{i2} = 1$ and $X_{i1} + X_{i3} = 1$ (one below the other). If X_{i1} corresponds to $\overline{Y}_i$ and X_{i2}, X_{i3} correspond to Y_i, then we also have the same pair of consecutive constraints. Note that in the first block of constraints there is a consecutive 1s matrix (each variable has either a single 1 or a pair of 1s and, if there is a pair of 1s, then these appear in two consecutive rows).

The second block of constraints in I' corresponds to satisfying the clauses, where for each clause the sum of the clause's variables is required to be at least one. So, in the second block of constraints, each variable (of I') appears exactly once, and therefore the coefficients' matrix for the second block is also a consecutive 1s matrix.

We now argue that the resulting set of inequalities in binary variables – integer programming – has a feasible solution if and only if I can be satisfied. To see this claim note that, given a truth assignment that satisfies I, we set $X_{ij} = 1$ if and only if its corresponding literal has a true value, and otherwise $X_{ij} = 0$. Then, the first set of constraints clearly holds because, if X_{ij} corresponds to Y_i and $X_{ij'}$ corresponds to $\overline{Y}_i$, then exactly one of X_{ij} and $X_{ij'}$ is set to 1, whereas the other one is set to 0. To see that the second set of constraints holds, note that at least one literal of each clause has a true value and thus the sum of their corresponding variables (of the Integer Programming) is at least 1. Given an integer solution to the Integer Programming, we assign a true value for each literal that its corresponding variables in I' have a unit value and otherwise (they are zero) we set them a false value. This is a truth assignment as, by the first set of constraints, we cannot assign a common value to both Y_i and $\overline{Y}_i$. The truth assignment satisfies all the clauses because of the second block of constraints.

To cast the Integer Programming for R3-SAT into a 2-multi-shift instance, we replace the equations by covering inequalities and we penalize strict inequalities in the objective function. More precisely, we define an objective function to be the sum of all variables. We replace the equations (that belong to the first set of constraints) by covering (inequality) constraints. Then, in the resulting 2-multi-shift instance, there is a feasible solution whose cost is at most the number of constraints in the first block of constraints (i.e., $2n$) if and only if there is a feasible solution to the Integer Programming instance. Therefore, the 2-multi-shift problem is NP-hard. ■

The following remark follows by considering the proof of Theorem 1.

Remark 2. The 2-multi-shift scheduling problem is NP-hard even if the right-hand side's coefficients are all equal to 1.

3.2. A simple k-approximation algorithm for the k-shift problem

We call the approximation algorithm that we present for the k-multi-shift problem the k-split algorithm, as it "splits" each column to up to k columns with consecutive 1s in each. Let a column i have $k_i \leq k$ blocks of consecutive 1s and replace the corresponding variable x_i by k_i variables $x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(k_i)}$, each with the same objective coefficient c_i. Each variable $x_i^{(j)}$ corresponds to a column of coefficients with 1s only in the rows that participate in the jth block of consecutive 1s in the original ith column. For every j, let the upper bound of $x_i^{(j)}$ be the upper bound of x_i. This completes the split process and results in an MCB instance with consecutive 1s in each of its $\sum_{i=1}^{n} k_i$ columns. Denote this new MCB instance by MCB$^{(k)}$, which, as an instance of a problem on consecutive 1s, can be solved optimally in polynomial time.

Let $\{\mathbf{x}_i^{(j)}\}_{i,j}$ be an optimal solution of MCB$^{(k)}$. The k-split algorithm returns the solution $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ where $\mathbf{x}_i = \max_{1 \leq j \leq k_i} \{\mathbf{x}_i^{(j)}\}$.

We note that the solution $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ is feasible. Moreover, each feasible solution $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ for an MCB of cost $\mathbf{C}$ has a corresponding feasible solution $\{\mathbf{x}_i^{(j)}\}$ for MCB$^{(k)}$ with a cost of at most $k\mathbf{C}$, and each feasible solution $\{\mathbf{x}_i^{(j)}\}$ for an MCB$^{(k)}$ with a cost of at most $\mathbf{C}$ has a corresponding feasible solution $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ for an MCB with a cost of at most $\mathbf{C}$. Thus, we conclude:

Observation 3. *The k-split algorithm is a k-approximation algorithm.*

Recall that a cyclical matrix is a k-multi-shift with $k = 2$. Since the time complexity of the k-split algorithm is $O(kn \log(m)(kn + m \log(m)))$, using Orlin's [21] min-cost network-flow algorithm for a network with m nodes and kn edges, we obtain the following corollary:

Corollary 4. *The k-split algorithm, when applied to MCB on cyclical matrices $\mathbf{A}$, is an $O(n \log m(n+m \log m))$-time 2-approximation algorithm.*

The same 2-approximation algorithm applies both for the 2-shift problem and the cyclical scheduling problem, yet the cyclical scheduling problem is a special case of the 2-shift problem where the first interval of 1s begins in row 1 and the second interval of 1s ends at row m. Because of this specific structure, it is potentially possible to get an improved approximation for the cyclical scheduling and perhaps even a polynomial time algorithm for solving it optimally. However, as we prove in Theorem 1, the 2-shift problem is NP-hard and thus it is unlikely that there is a polynomial time algorithm that solves the problem optimally.

3.3. The complexity of the cyclical scheduling problem

Despite the importance of cyclical scheduling problems, little is known about the complexity of solving the multicover problem with cyclical 1s in columns. There is, however, some related research that can be linked to this problem, as we discuss next.

Hochbaum and Tucker [12] solved the set cover problem on cyclical 1s matrices in polynomial time $O(m(n + m \log m))$. They noted that the complexity of the multicover problem on cyclical matrices has not been established.

One problem related to the cyclical MCB is the *exact matching* problem. This problem is a perfect matching problem defined on a graph with a subset of the edges painted blue. The problem is to find a perfect matching that contains exactly k blue edges. This problem is known to be solved in randomized polynomial time and belongs to the complexity class RNC (it thus belongs to the class RP) by an algorithm proposed by Mulmuley et al. [20]. This implies an RP algorithm also for the bipartite version of the problem, which in turn implies an RP algorithm for solving the exact bipartite flow problem that has exactly k blue arcs with positive flow. To date, the complexity status of these problems was not established to be either in P or NP-hard. We note that the problem of finding a unique perfect matching in a bipartite graph, if such exists, is in NC as was shown in [16]. Another relevant result on the bipartite exact matching is that, if the input graph is a complete bipartite graph, then the problem can be solved in polynomial time (see [14,25]). The following theorem shows that MCB on cyclical matrices is at least as difficult as the exact bipartite matching problem.

Theorem 5. *Assume that there is a polynomial time algorithm that solves (non-uniform) MCB on cyclical matrices. Then, there is a polynomial time algorithm that solves the exact bipartite matching problem.*

Proof. Consider an instance of the *exact bipartite matching problem*, $G = (S, T, E_b \cup E_r)$ and k, i.e., we are given a bipartite graph G with a partition of the nodes to two sides S and T, and a partition of the edge set into blue edges E_b and red edges E_r. The question is whether there is a perfect matching in G with exactly k blue edges. We assume that we have a polynomial time algorithm for MCB when the constraint matrix A is cyclical, and we show how to solve the exact bipartite matching using it. We assume that the algorithm for MCB is for the formulation without the surplus variables and using covering inequalities.

Consider the following auxiliary feasibility problem EM (exact matching):

$$\text{(EM)} \quad \text{Min} \quad 0^t x \tag{1}$$

subject to :

$$\sum_{i \in S} x_{ij} = 1 \quad \forall j \in T \tag{2}$$

$$\sum_{j \in T} -x_{ij} = -1 \quad \forall i \in S \tag{3}$$

$$\sum_{(i,j) \in E_b} x_{ij} = k \tag{4}$$

$$0 \le x_{ij} \le 1 \quad \forall (i, j) \in E_b \cup E_r \tag{5}$$

$$x_{ij} = 0 \quad \forall (i, j) \notin E_b \cup E_r. \tag{6}$$

An integer solution to EM is a perfect matching, as enforced by constraints (2) and (3). Constraint (4) ensures that any feasible matching includes exactly k blue edges. Therefore, a polynomial time algorithm that solves EM in integers is a polynomial time algorithm for the exact bipartite matching problem.

It remains to show how to transform EM to an MCB instance with a cyclical constraint matrix. W.l.o.g. we sort the rows of EM so that there first appear the rows that correspond to S and then those rows that correspond to T. In the first step, we multiply each column of the constraint matrix of EM that corresponds to a red edge by -1 by substituting $y_{ij} = 1 - x_{ij}$ for each red edge, and then updating the right-hand side (by moving all the 1s from the above transformation to the right-hand side, and keeping only variables on the left-hand side). This substitution does not affect the lower and upper bounds for each variable. Ignoring the bound constraints, this forms a constraint matrix that has, for every red edge's column, only one 1 and one -1, and the row that contains the 1 is below the row that contains the -1, i.e. consecutive column type. Also, every blue edge's column has a -1 that is below one 1 and there is another 1 that appears in the complicating constraint—a format of cyclical column type.

Since each column of the matrix has one of the three types of columns as in a cyclical matrix transformed by T, when we apply the inverse transformation T^{-1} it results in a cyclical matrix. So, in the second step we apply the transformation T^{-1}. We obtain a new *feasibility problem* $\min 0^t x$ subject to $A'x = b'$ and $x_j \in \{0, 1\}\ \forall j$, where A' is cyclical. The feasibility problem can be posed as an MCB problem by replacing each equation with a cover inequality, and penalizing the strict inequalities in the objective function. By penalizing the strict inequalities, we mean that the objective function is set to be $\min \sum_i \left(\sum_j A'_{ij} x_j - b'_i \right)$. The feasibility problem has a solution if and only if the above MCB instance has a zero cost feasible solution. ■

Although the complexity status of a multicover problem with a cyclical constraint matrix is open, the uniform requirements case is solvable in polynomial time, as we see next.

4. Solving multicover problem with cyclical 1s and uniform right-hand side

In this section, we provide a polynomial time algorithm that solves the multicover problem MC when the constraint matrix is cyclical, and the right-hand side b is uniform i.e., $b_i = B, \forall i$.

We first duplicate the first constraint as the (new) last constraint. The resulting problem is clearly equivalent to the original one. Next, we subtract the surplus variables from each inequality, and we assume that A is the constraint

matrix after this transformation. The duplication of the first constraint is done before subtracting the surplus variables. Moreover, by the duplication of the first constraint, A does not have a consecutive column with 1 in the last row (however, our transformation may make a consecutive 1s column into a cyclical column). So, 1 in the last row means that the column is cyclical. The matrix A is assumed to also contain the columns of the surplus variables, which contain a single -1 entry (this is the column of the last surplus variable). The surplus variables are denoted as part of the vector x by setting $x_{n+i} = s_i$ for $i = 1, 2, \ldots, m$. The constraints are thus given by $Ax = b$, $x \geq 0$.

Note that, in the transformed matrix TA, each column has at most a single 1 in the first $m-1$ rows and at most a single -1 in the first $m-1$ rows. The only case where there is no 1 in the first $m-1$ rows of a given column is when this column corresponds to a surplus variable s_1 (this is so because the last row equals the first row and therefore each consecutive 1s column has a 0 in the last row and therefore it is transformed into a column with a single 1 that is in the first $m-1$ rows. The only case where there is no -1 in a column is where the corresponding column of A is equal to the vector of 1s.

The outline of our method is as follows. Using the transformation T, we transform MCB into an auxiliary problem that is a network flow problem with an additional constraint. This constraint is that the amount of flow along *good edges* is at least B, where the good edges correspond to columns with an entry 1 in the last row. This is done as in Section 2. To solve this auxiliary problem, we will compute for each integer value $k = 1, \ldots, n-1$ the minimum cost cycle that contains at least k good edges. Then, we will compute the solution to the problem of selecting a collection of cycles of minimum total cost that contain at least B good edges, by presenting it as a polynomially solvable Knapsack problem.

Lemma 6. *The auxiliary problem is equivalent to a problem with:*

(a) *all objective function coefficients are strictly positive;*

(b) *all rows of the constraint matrix, except for the last one, have at least one 1 and one -1;*

(c) *there is at most one column that has one 1 in the last constraint and all other entries 0.*

Proof. (a) If there is a j such that $c_j < 0$, then the optimal solution is unbounded with $x_j = \infty$. If $c_j = 0$, then, setting $x_j = \infty$, the remaining problem is a multicover problem with cyclical 1s in columns with a smaller subset of constraints that are still unsatisfied—those that have 0 in the column of x_j. Therefore, we can assume that $c_j > 0$ for all j.

(b) We denote by $A' = T \cdot A$ the transformed matrix that results from Veinott and Wagner's transformation. Note that, in A', the columns that correspond to surplus variables also have one 1 and one -1 per column. Also, the right-hand side vector after the transformation is a vector with a single non-zero component in the last row that equals B. We denote by A'' the matrix obtained from A' by deleting the last row. In A'', every column beside that corresponding to s_1 has at most a single -1 and exactly a single 1. To see this last property of A'', note that a column that corresponds to a surplus variable s_k has 1 in row $k-1$ that is in the first $m-1$ rows of the transformed matrix, a cyclical column that corresponds to x_j has a single 1 in the first $m-1$ rows since the first block of 1s ends before row $m-1$, and a consecutive column that corresponds to x_j has a zero in the last row in A, and therefore it has a single 1 in the first $m-1$ rows of A'. Moreover, each column in A' that does not have a -1 entry must be generated by a column of A that has all its entry equal to 1. Then, the entry in A' that equals 1 is in the last row. For row i of A', denote by P_i the set of variables whose coefficient in row i is 1, and by N_i the set of variables whose coefficient in row i is -1.

We now apply a preprocessing procedure. If a row $i < m$ has $N_i = \emptyset$, then we delete it from the matrix. If a row i has $P_i = \emptyset$, then every variable in N_i must equal 0, so we delete their columns from the matrix, and then we delete row i from the matrix. The resulting problem is equivalent to the original one.

(c) Consider the set of columns $C0 = \{j | a'_{m,j} = 1 \text{ and } a'_{i,j} = 0 \ \forall i < m\}$. Because the variables are unbounded, then among all the columns of $C0$ there is at most one that participates in an optimal solution—the one that has minimum cost among all $C0$. ■

Next, we remove from A' and A'' the columns that have only -1 entry and no 1 entry. This is done by replacing the equation constraints by an inequality covering constraints. This new formulation is a relaxation of the original one, however the solution for the new problem that we will construct satisfies all these constraints as equalities, and hence this does not change the problem.

Let A' and A'' be the matrices after the preprocessing described in the lemma. Then, every row of A'' has some -1s and some 1s, and the set of constraints of the transformed problem is:

$$\sum_{j\in P_i} x_j - \sum_{j\in N_i} x_j \geq 0 \quad \forall i < m$$
$$\sum_{j\in P_m} x_j \geq B,$$

where the first set of constraints is $A''x \geq \mathbf{0}$ (where $\mathbf{0}$ is the vector whose entries are all equal 0).

Consider the set of columns that have both 1 and -1 in A''. Construct a directed multi-graph $G = (V, E)$ over a vertex set $\{1, 2, \ldots, m-1\}$ that corresponds to the set of constraints in A'' as follows: there is a directed edge from p to q if there is a column j in A'' with 1 in row p and -1 in row q. For edge (p, q) corresponding to column j, the cost of the edge is set to $C_{p,q} = c_j$.

An edge (p, q) of G that corresponds to column j in A'' is *a good edge* if $j \in P_m$, and *a bad edge* otherwise.

Let $c_{j'} = \min\{c_j | j \in C0\}$. We add to G a new vertex j' that has only two loops that are incident to it (and no other incident edges): one of them is a good edge with cost $c_{j'}$ and the other one is a bad edge with zero cost. Both these loops correspond to the j'th column.

We use the Floyd–Warshall algorithm to compute the shortest path distance matrix, where $C'_{p,q}$ is the cost of path $P^b_{p,q}$ from p to q in G using only bad edges.

Next we construct a new directed bipartite graph $G' = (S \cup T, E')$, where $S = \{v_s | v \in V\}$ and $T = \{v_t | v \in V\}$. E' and a cost D of each edge are defined as follows:

$$E' = E_g \cup E_b$$
$$E_g = \{(p_s, q_t) | (p, q) \text{ is a good edge}\} \quad \text{and} \quad D_{p_s,q_t} = C_{p,q}$$
$$E_b = \{(q_t, p_s) | p, q \in V\} \quad \text{and} \quad D_{q_t,p_s} = C'_{q,p}.$$

Since G' is a bipartite graph, then every cycle in G' has even length. Moreover, in every closed walk in G', exactly half of its edges are good edges.

Next, we consider multi-graphs over $V' = S \cup T$. We say that two multi-graphs are *disjoint* if they do not use the same copy of the same edge. The next two lemmas establish the equivalence between disjoint cycles in G' that contain at least B edges, and a feasible solution to the multicover problem.

Lemma 7. *Every collection of disjoint cycles over $S \cup T$ that contains at least B good edges corresponds to a feasible solution to the multicover problem of the same cost.*

Proof. We initialize the value of all the variables to zero and process all the edges in the collection as follows:

For a good edge, we increase the value of the corresponding variable by the number of copies that appear in the collection.

For a bad edge (q_t, p_s) such that $q \neq p$, its cost is the shortest distance from q to p in G. For every edge in $P^b_{q_t,p_s}$, we increase the corresponding variable by the number of copies that (q_t, p_s) appears in the collection.

By construction, the obtained solution has the same cost as the collection of cycles. The collection of cycles contains at least B good edges and therefore $\sum_{j\in P_m} x_j \geq B$. Since, for every vertex, the in-degree equals out-degree, then, for every row $i < m$, $\sum_{j\in P_i} x_j = \sum_{j\in N_i} x_j$.

Therefore, the ith constraint is satisfied. Therefore, the resulting solution is feasible. ■

Lemma 8. *Every feasible solution to the multicover problem corresponds to a collection of disjoint cycles in G that contains at least B good edges with the same cost.*

Proof. For every j, we take into the edge multi-set of the collection x_j copies of the edge that corresponds to the jth column. This is done as follows:

- If $a''_{p,j} = 1$ and $a''_{q,j} = -1$, we add x_j copies of the edge (p, q).
- If $a'_{m,j} = 1$ and $a''_{i,j} = 0\ \forall i$, then we add x_j copies of the good loop.

In the resulting edge multi-set for every vertex, its out-degree equals its in-degree. Therefore, the edge multi-set can be partitioned into disjoint closed walks (every connected component is Eulerian). ■

Remark 9. A closed walk can be partitioned into a set of simple cycles.

We note that, in G', every cycle has an even number of edges and exactly half of them are good edges. By Lemmas 7 and 8 and Remark 9, in order to find an optimal multicover, we can find an optimal cover of G' with a collection of closed walks with at most $|V'|$ edges each, such that the total number of copies of good edges is at least B.

In order to do so, we compute $D^k_{i,j}$, the length of a shortest walk from i to j that uses exactly k edges (for every $i, j \in V'$ and $k \leq |V'|$). Then, $D^1_{i,j} = D_{i,j}$ and $D^k_{i,j} = \min_l\{D^{k-1}_{i,l} + D_{l,j}\}$. Next we compute C^k, the length of a shortest closed walk that contains exactly k good edges by $C^k = \min_{i,j}\{D^{2k-1}_{i,j} + D_{j,i}\}$.

The remaining problem is formulated as the following Knapsack problem:

$$\text{Min} \sum_j C^j X_j \tag{7}$$

subject to :

$$\sum_j j X_j \geq B$$

$$X_j \geq 0 \text{ integer.}$$

In this formulation, X_j is the number of copies of the shortest closed walk with exactly j good edges. This formulation is a Knapsack problem which is special in that the coefficients in the constraint are small. In the following lemma, we show how to solve such a Knapsack problem.

Lemma 10. *Problem (7) can be solved in $O(n^3 \log B)$ time.*

Proof. Consider a scaling parameter s. Let the scaled problem be

$$\text{Min} \sum_j C^j X_j \tag{8}$$

subject to :

$$\sum_j j X_j \geq \frac{B}{s}$$

$$0 \leq X_j \leq \left\lceil \frac{B}{s} \right\rceil$$

$$X_j \text{ integer.}$$

Let the optimum for (7) be x^* and the optimum for (8) be $x^{(s)}$. Let Δ be the largest sub-determinant of the constraint matrix and in our problem $\Delta = n$.

Hochbaum and Shanthikumar [11] introduced the *proximity-scaling algorithm* for convex minimization over linear constraints that makes use of the following *proximity theorem* (see Theorem 3.5 in [11]): for all j, $|x^*_j - x^{(s)}_j s| \leq ns\Delta$. In our case, this bound is $n^2 s$.

We now set $s \leftarrow B/(4n^2)$, and we scale the problem by s. The right-hand side in (8) is at most $4n^2$, so we can solve the resulting problem using the standard dynamic programming procedure in $O(n^3)$ steps. From the proximity theorem, the updated lower and upper bounds on each variable are:

$$\max\{0, x^{(s)}_j s - n^2 s\} \leq x_j \leq x^{(s)}_j s + n^2 s.$$

We now set $s \leftarrow \frac{s}{2}$. In the new scaled problem, each variable has a range of $4n^2$ units at most. Note that $\sum_i i x^{(s)}_i s \leq B + ns$, as otherwise at least one $x^{(s)}_i$ could have been decremented—leading to a better solution. By substituting for each variable its lower bound, the right-hand side is $2(n^2 s + ns)$ units at most.

So, again, we have a scaled Knapsack problem with a right-hand side equal to $4n^2$, at most, that can be solved in $O(n^3)$. This process is repeated $O(\log B)$ times, to solve the problem in $O(n^3 \log B)$ steps. ■

Therefore, the total time-complexity of our algorithm for the multicover problem with uniform right-hand side is $O(n^3 \log B + n^4)$. We conclude this result with the following theorem:

Theorem 11. *Problem MC with a cyclical constraint matrix and uniform requirements* ($b_i = B \ \forall i$) *is solvable in* $O(n^3 \log B + n^4)$.

5. Concluding remarks

We first note that, for the cyclical multicover problem where the right-hand side varies, the corresponding polyhedron has at most one fractional extreme point. In some special cases as used in [4], it is sufficient to use one cut to dispose of this fractional extreme point. It is left as an open question to try to extend this result to a more general case.

In some cases, the organization of a work schedule is such that a collection of work periods do not form temporally consecutive time periods.

In the airline industry, the problem of crew scheduling is a prominent one. Its management has a major impact on overall costs, as shown by Barnhart et al. [3]. A *pairing* is a sequence of duty periods with intervening overnight rests and beginning and ending at a crew base. *Reserve* crew schedules consist of groups of consecutive on-duty and off-duty days, typically lasting 30 days. The pattern of reserve crew schedules has to abide by legal and union regulations as well as being practically feasible. Reserve patterns dictate the rules by which reserve schedules can be generated. These assume the form of the length of each on-duty block of consecutive periods and rules on the amount of separation of such blocks by off-duty periods.

In the retail industry, scheduling labor is not primarily a question of which days of the week to choose, but rather the choice of shift. If a worker is assigned to six days per week, all these days will have the same shift, so all are morning, afternoon or evening shifts. In scheduling police force, each consecutive on-duty block is all day shift, evening shift or night shift.

If the time periods are ordered temporally, then an afternoon shift of a certain day precedes an evening shift of the same day. Ordering each day temporally, we get per day an M, A, E triplet. Thus, for example, a work schedule of four days of evening shift (E) from Monday through Thursday on a schedule of a week beginning on Monday appears as

$$(0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0).$$

Here we have four blocks, with each block consisting of a single 1. An approach that clusters the work schedule together in blocks is to arrange the schedule according to shifts. So, a work schedule will be a characteristic vector in the order,
$(Mon-M, Tue-M, Wed-M, Thu-M, Fri-M, Sat-M, Sun-M, Mon-A, Tue-A, Wed-A, Thu-A, Fri-A, Sat-A, Sun-A, Mon-E, Tue-E, Wed-E, Thu-E, Fri-E, Sat-E, Sun-E)$.

In the above example, with this ordering we have a single block of consecutive 1s. Therefore, it is desirable to arrange work schedules so that the same shifts within the planning horizon appear consecutively, even though there are other intermediate shift periods that appear according to the time line order.

Given a multi-covering instance, we can look for a rearrangement of the rows in order to minimize the maximum number (among all columns) of blocks of consecutive 1s in the column. This rearrangement is motivated by the k-approximation algorithm for the k-multi-shift scheduling problem. This argument motivates the study of the problem of finding the best rearrangement, i.e., to classify its complexity and to design good algorithms to solve it. We leave this for future research.

Acknowledgements

The authors wish to express their gratitude to anonymous referees whose comments and suggestions improved and simplified the presentation of the results in this paper.

The first author's research was supported in part by the US National Science Foundation (NSF) awards DMI-0085690 and DMI-0084857.

References

[1] U. Adamy, C. Ambuehl, R.S. Anand, T. Erlebach, Call control in rings, in: Proceedings of ICALP 2002, in: LNCS, vol. 2380, 2002, pp. 788–799.
[2] M.J. Atallah, D.Z. Chen, D.T. Lee, An optimal algorithm for shortest paths on weighted interval and circular-arc graphs, with applications, Algorithmica 14 (1995) 15–26.
[3] C. Barnhart, E.L. Johnson, G.L. Nemhaurser, P. Vance, Crew scheduling, in: R.W. Hall (Ed.), Handbook of Transportation Science, Kluwer Academic Publishing, Boston Mass, 1999, pp. 492–521.
[4] J.J. Bartholdi, J.B. Orlin, H.D. Ratliff, Cyclic scheduling via integer programs with circular ones, Operations Research 28 (1980) 1074–1085.
[5] J.J. Bartholdi, A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering, Operations Research 29 (1981) 501–510.
[6] V. Chvátal, A greedy heuristic for the set-covering problem, Mathematics of Operations Research 4 (1979) 233–235.
[7] G. Dobson, Worst-case analysis of greedy heuristics for integer programming with non-negative data, Mathematics of Operations Research 7 (1982) 515–531.
[8] T. Erlebach, private communication, 2003.
[9] M.R. Garey, D.S. Johnson, Computers and Intractability, Freeman, San Francisco, 1979.
[10] N.G. Hall, D.S. Hochbaum, A fast approximation algorithm for the multicovering problem, Discrete Applied Mathematics 15 (1986) 35–40.
[11] D.S. Hochbaum, J.G. Shanthikumar, Convex separable optimization is not much harder than linear optimization, Journal of the ACM 37 (1990) 843–862.
[12] D.S. Hochbaum, P.A. Tucker, Minimax problems with bitonic matrices, Networks 40 (2002) 113–124.
[13] D.S. Hochbaum, Approximation algorithms for the set covering and vertex cover problems, SIAM Journal on Computing 11 (1982) 555–556. an extended version: W.P. #64-79-80, GSIA, Carnegie-Mellon University, April 1980.
[14] A.V. Karzanov, Maximum matching of given weight in complete and complete bipartite graphs, Kibernetika 1 (1987) 7–11.
[15] G.J. Koop, Multiple shift workforce lower bounds, Management Science 34 (1988) 1221–1230.
[16] D. Kozen, U.V. Vazirani, V.V. Vazirani, NC algorithms for comparability graphs, interval gaphs, and testing for unique perfect matching, in: Proc. FSTTCS 1985, in: LNCS, vol. 206, 1985, pp. 496–503.
[17] H.C. Lau, Combinatorial approaches for hard problems in manpower scheduling, Journal of the Operations Research Society of Japan 39 (1996) 88–98.
[18] C. Lund, M. Yannakakis, On the hardness of approximating minimization problems, Journal of the ACM 41 (5) (1994) 960–981.
[19] J.G. Morris, M.J. Showalter, Simple approaches to shift, days off and tour scheduling problems, Management Science 29 (1983) 942–950.
[20] K. Mulmuley, U.V. Vazirani, V.V. Vazirani, Matching is as easy as matrix inversion, Combinatorica 7 (1987) 105–114.
[21] J.B. Orlin, A faster strongly polynomial minimum cost flow algorithm, Operations Research 41 (1993) 338–350.
[22] A. Schrijver, Combinatorial optimization polyhedra and efficiency, Springer-Verlag, Berlin, 2003.
[23] A.F. Veinott, H.M. Wagner, Optimal capacity scheduling: Parts I and II, Operations Research 10 (1962) 518–547.
[24] R.V. Vohra, A quick heuristic for some cyclic staffing problems with breaks, Journal of Operational Research Society 39 (1988) 1057–1061.
[25] T. Yi, K.G. Murty, C. Spera, Matching in colored bipartite networks, Discrete Applied Mathematics 121 (2002) 261–277.