アルゴリズム 37－4
（1994．1．25）

人的資源スケージューリング問題の難しさについて

ラウ・フン・チュイン

東京工業大学 理工学研究科
情報工学専攻 渡辺研究室

目黒区大岡山2-12-1

hclau@cs.titech.ac.jp

資源管理の一つとして重要な問題である人的資源のスケジューリング問題に対し，その計算の複雑さについて議論する．ＮＰ困難な人的資源スケジューリング問題のクラスや多項式時間計算可能な問題のクラスを示す．

On the Complexity of Manpower Shift Scheduling

LAU Hoong-Chuin

Department of Computer Science,
Tokyo Institute of Technology

Meguro-ku Ookayama 2-12-1, Tokyo 152, Japan

hclau@cs.titech.ac.jp

We consider the problem of shift assignment in manpower scheduling. We enumerate 8 variants of the problem: the schedule may be normal or cyclic, the shift change constraint may be monotonic or unrestricted, and the demands may be exact or slack. We show the NP-completeness of four of them, and present polynomial algorithms to solve two others. The complexity of two other variants remains open. Our work formally defines the computational intractibility of manpower shift scheduling and thus justifies existing works in developing manpower scheduling systems using combinatorial and heuristic techniques.

1 Introduction

Manpower scheduling is a critical operations management activity in large manufacturing or service organizations which operate round-the-clock. It is concerned with the scheduling of workers to work on shifts to meet operational requirements in ways that satisfy the constraints imposed by the management, labour union and the government. The manpower scheduling problem is well studied in operations research. Recently, Tien and Kamiyama [TK82] presented a systemetic framework for solving the general manpower scheduling problem. Since then, various manpower scheduling methods have been proposed, such as integer programming [KJ91], heuristics [BK87], and network optimization [BW90].

The underlying problem in manpower scheduling is the shift scheduling problem [GM86] which determines the shift patterns assigned to workers subject to scheduling policies such as those descrived above. In this paper, we consider a restricted version of the shift scheduling problem, which we term the *Constraint Shift Assignment Problem* or CSAP. The problem is concerned with finding a satisfying assignment of shifts to workers such that demands are met and the so-called shift change constraints are not violated. We study 8 variants of the problem with respect to the nature of the schedule, the property of the shift change constraints, and the property of the demand matrix. In this paper, we present NP-completeness results of 4 variants and propose greedy algorithms to solve 2 other variants. The computational complexity of 2 remaining variants remains open.

2 Preliminaries

A *worker* is a unit of manpower. A *planning period* is the number of days for which manpower scheduling is done. A *shift* is a period of time within a day for which a worker turns out for work. Otherwise, the worker gets an *offday*. We assume different shifts to be numbered 1,2,... and conveniently use 0 to denote an offday. A *schedule*, is an assignment of shifts and offdays to individual workers over days of the planning period. Each element of the schedule is known as a *slot*. We assume that initially offdays are already preassigned to the schedule. A *demand matrix* shows, for each day and each shift, the number of workers required. Figure 1(a) gives an example of a demand matrix; 1(b) gives an example of an initial schedule; and 1(c) shows a schedule which satisfies the demand. A *workstretch* refers to the contiguous working days between two offdays. The *start day* of a workstretch refers to its first working day. *Shift change constraints* define what shifts may follow a shift of any two adjacent slots. As we assume that an offday may precede or follow any shift, shift change constraints need only be considered within workstretches. A *feasible schedule* is one with all its slots instantiated to shifts which meets the demand requirements and satisfies the shift change constraints. Each row of the feasible schedule is known as a *shift pattern*.

2.1 Problem Definition

CSAP is defined as, given the planning period, number of workers, number of shifts, demand matrix and shift change constraints, find a feasible schedule. Let K' = number of workers, I = planning period, J = number of shifts. Let D = $I \times J$ demand matrix, where $D_{i,j}$ represents the demand units of day i shift j, for $i = 1, ..., I$ and $j = 1, ..., J$. Let δ = shift change matrix which defines the shift change constraints, i.e., $\delta_{j_1,j_2} = 1$ if shift j_2 may follow shift j_1 and 0 otherwise, for $j_1, j_2 = 1, ..., J$. Let $S = K' \times I$ schedule preassigned with offdays. As shift change constraints apply only within workstretches, from S, we construct a workstretch-based schedule σ by having each workstretch occupy one row. For example Figure 1(d) is the σ derived from Figure 1(b). Let K = number of workstretches in σ, σ_k (or simply

k) denote the kth workstretch, $\sigma_{k,i}$ denote the slot on position (i.e. day) i of σ_k, and $|\sigma_k|$ denote the length (number of slots) of workstretch k. Thus, an input instance of CSAP is given by a 6-tuple $(I, J, K, D, \delta, \sigma)$. The output is a feasible schedule σ^*. CSAP is an NP search problem. That is, it corresponds to the following decision problem that is in NP:

INSTANCE: $(I, J, K, D, \delta, \sigma)$.

QUESTION: Is there a feasible schedule?

When discussing NP-completeness, we consider decision problems like above and refer them also as CSAP. Clearly, the corresponding searching problems are harder than decision problems.

2.2 Variants of CSAP

We study several popular variants of CSAP. Firstly we consider two types of schedules: normal and cyclic schedules. A *cyclic schedule* is one in which shift patterns rotate across workers operationally, i.e. after I days, a worker k would take the shift pattern of worker $k+1$, for $k = 1, ..., K'-1$, and worker K' would wraparound to take the shift pattern of worker 1. Hence, shift change across shift patterns will have to be considered. For this purpose, two adjacent workstretches will be concatenated as one if there are no offdays between the last working day of one and the first working day of the other. Figure 1(e) is the workstretch-based cyclic schedule corresponding to Figure 1(b).

Secondly, we consider the case that the total number of workers not having offdays on a given day exceeds the column sum of demands for all shifts on that day. We call this a situation of *slack demand*, as opposed to *exact demand*. Slack demand units will be satisfied by the assignment of a special placeholding shift called the $*$-shift to some workers.

Finally, we consider the *monotonic* shift change property. In manpower scheduling, it is a common practice that, as the days progress, a worker's should be turned up for work no earlier than the day before so that he maintains a healthy biological clock. Assuming that we order shifts according to their start times, then shifts should be assigned in non-decreasing order. This non-decreasing sequence may be broken by an offday because workers are expected to get enough rest during the offday to be turned up for an earlier shift next. In terms of CSAP, this means that the monotonic shift change matrix is upper-triangular with all 1's. Where monotonicity is not required, we refer to the shift change as being *unrestricted*.

Hence, we define 8 variants of CSAP in terms of a three-field classification $\alpha|\beta|\gamma$, where,

1. α = Normal/Cyclic Schedule (N/C)
2. β = Monotonic/Unrestricted Shift Change Matrix (M/U)
3. γ = Exact/Slack Demands (E/S)

For instance, CSAP$(N|U|E)$ refers to CSAP with normal schedule, unrestricted shift change matrix and exact demands. Clearly, the normal schedule is a special case of the cyclic schedule, the monotonic shift change is a special case of the unrestricted shift change, and the exact demands is a special case of the slack demands. Thus, if we prove that the special case is NP-hard, then the corresponding general case will also be NP-hard.

3 NP-Completeness

We will show the completeness of CSAP$(N|U|E)$ by a polynomial many-one reduction from 3SAT. Note that throughout this section, we consider only decision problems.

Let $U = \{U_1, U_2, ..., U_n\}$ be a set of *Boolean variables*. A *truth assignment* for U is a function $t : U \rightarrow \{True, False\}$. If X is a variable in U, then X and $\overline{X}$ are *literals* over U. A *clause* C over U is a disjunction of literals over U, and we say that C is *satisfied* by a truth assignment if at least one of its members is $True$ under that assignment. A *3CNF* over U is represented by a set of clauses

$F = \{C_1, C_2, ..., C_m\}$ over U with three literals per clause. We say that F is *satisfiable* if there exists a truth assignment for U that simultaneously satisfies all clauses in F. Now the problem *3SAT* is defined as follows [GJ79]:

INSTANCE: A set U of boolean variables $\{U_1, U_2, ..., U_n\}$ and a 3CNF represented by $F = \{C_1, C_2, ..., C_m\}$ over U.

QUESTION: Is there a truth assignment for U such that F is satisfiable?

Lemma 3.1. For any 3CNF F, there exists a 3CNF F' of m clauses such that

1. F' is satisfiable iff F is satisfiable,
2. for every $i \in \{1, ..., n\}$, $p_i = \overline{p_i}$, where p_i (resp., $\overline{p_i}$) is the number of occurrences of U_i (resp., $\overline{U_i}$) in F'. and
3. $m \leq P$, where $P = \sum_{i=1}^{n} p_i$.

Proof. For any $i \in \{1, ..., n\}$, let q_i ($\overline{q_i}$) denote the number of occurrences of U_i ($\overline{U_i}$) in F. If $q_i \geq \overline{q_i}$, we add clauses $C_{m+1} = C_{m+2} = \cdots = C_{m+q_i-\overline{q_i}} = (\overline{U_i} \vee U_1 \vee \overline{U_1})$. Likewise, if $q_i < \overline{q_i}$, we add clauses $C_{m+1} = C_{m+2} = \cdots = C_{m+\overline{q_i}-q_i} = (U_i \vee U_1 \vee \overline{U_1})$. The desired F' is F plus these additional clauses. Clearly, these additional clauses do not affect the satisfiability of F, thus F' is satisfiable iff F is satisfiable. Furthermore, $p_i = \max\{q_i, \overline{q_i}\}$, and each variable U_i occurs $2p_i$ times (p_i times for each of U_i and $\overline{U_i}$) in F'. Since each clause has three literals, $3m = 2P$; hence $m \leq P$. □

Theorem 3.1. CSAP($N|U|E$) is NP-complete, even for fixed $I \geq 5$.

Proof. Let F be an instance of 3SAT. In the following, we assume that F is already converted by Lemma 3.1. U, F, n, m, p_i, and P are defined as above. We construct an instance of CSAP as follows (in the following, the ranges of i and k are respectively $\{1, ..., n\}$ and $\{1, ..., m\}$):

1. $K = 2P$.
2. $I = 5$.
3. $J = 4P + m + 4$, where the shift numbers are labelled as follows:
   - $x_{ij}, \overline{x_{ij}}, v_{ij}, w_{ij}$, for all pairs of i and $j \in \{1, ..., p_i\}$,
   - c_k, for all k, and u, y, z, and θ.
4. The demand matrix is set as follows:
   - $D_{2,v_{ij}} = D_{2,w_{ij}} = D_{3,x_{ij}} = D_{3,\overline{x_{ij}}} = 1$, for all pairs of i and $j \in \{1, ..., p_i\}$,
   - $D_{1,u} = D_{4,y} = D_{5,z} = P$,
   - $D_{4,c_k} = 1$, for all k,
   - $D_{4,\theta} = P - m$, and
   - all the other demand matrix elements are set to 0.
5. Offdays are assigned within the schedule σ as follows:
   - $\sigma_{k,1} = 0$ (off) for $k = P + 1, ..., 2P$, and
   - $\sigma_{k,5} = 0$ (off) for $k = 1, ..., P$.
6. The shift change matrix δ is defined as follows (to improve readability, we adopt the notation $a \rightarrow b$ (read as shift a can be followed by shift b) iff $\delta_{a,b} = 1$):
   - for all pairs of i and $j \in \{1, ..., p_i\}$, $u \rightarrow v_{ij}$, $v_{ij} \rightarrow x_{ij}$, $v_{ij} \rightarrow \overline{x_{ij}}$, $x_{ij} \rightarrow y$, $\overline{x_{ij}} \rightarrow y$, $x_{ij} \rightarrow \theta$, $\overline{x_{ij}} \rightarrow \theta$, $w_{ij} \rightarrow x_{ij}$, and $w_{ij} \rightarrow \overline{x_{ij'}}$, where $j' = j - 1$ (if $2 \leq j \leq p_i$) and $j' = p_i$ (if $j = 1$);
   - for all i, k such that $j \in \{1, ..., p_i\}$, $x_{ij} \rightarrow c_k$ (resp. $\overline{x_{ij}} \rightarrow c_k$) if U_i (resp. $\overline{U_i}$) occurs the jth time in clause k;
   - $y \rightarrow z$; and
   - all other shift changes are set to 0.

Figure 2 depicts the CSAP instance associated with the following 3CNF F and truth assignment t:

$F = (U_1 \vee U_2 \vee U_3) \wedge (U_1 \vee \overline{U_2} \vee \overline{U_3}) \wedge$
$(\overline{U_1} \vee U_2 \vee U_3) \wedge (\overline{U_1} \vee \overline{U_2} \vee \overline{U_3})$;
$t(U_1) = True$; $t(U_2) = False$; $t(U_3) = True$;

We explain the motivation for the above construction. We call the top-half rows of a schedule σ (i.e. $\sigma_{k,i}$ for $k = 1, ..., P$) the *True Region* (TR) while the bottom-half, the *False Region* (FR). The demand matrix, offdays assignments, and shift change matrix force different shifts to occupy different regions in the schedule. Essentially, the shifts u's are used to force all v's to occupy TR, and thus w's to occupy FR. Likewise, the z's are used to force all c's and θ to occupy TR, and all y's to occupy FR.

Each $x_{ij}(\overline{x_{ij}})$ corresponds to the jth occurence of the literal U_i $(\overline{U_i})$ in F. The idea is to force all true literals occupy TR while the rest, FR. To do so, the following two conditions must be met:

(1) For any x_{ij} in TR, then $\overline{x_{ij}}$ is in FR, and vice versa.

(2) For any x_{ij} in TR, all other $x_{ij'}$'s ($j' \in \{1, ..., p_i\}$) are in TR. Same applies to $\overline{x_{ij}}$.

Condition (1) is made possible by the use of shifts v and w. For any i, j, since exactly one v shift and one w shift may be followed by x_{ij} and $\overline{x_{ij}}$, and that v's and w's reside in different regions, x_{ij} and $\overline{x_{ij}}$ must be in different regions. Condition (2) is enforced by having the seemingly strange shift constraint from w's to x's above. Suppose $\overline{x_{11}}$ is in FR, then w_{12} has to precede it; since x_{12} is preceded only by v_{12} or w_{12} and w_{12} is taken up, x_{12} cannot but reside in the TR. This implies that $\overline{x_{12}}$ must be in FR, and the argument repeats.

This completes the definition of our reduction. Clearly the above reduction can be done in polynomial time. We claim that F has a satisfying assignment if and only if the constructed CSAP instance has a feasible schedule.

First, assume F has a satisfying truth assignment t. Based on this t, we define a feasible schedule. For each $t(U_i) = True$, the schedule is defined so that the corresponding x_{ij}'s occupy TR while the $\overline{x_{ij}}$'s occupy FR. The reverse occurs for each $t(U_i) = False$. Since each clause has at least one true literal, every c_k residing in TR can be preceded by some x. Clearly, one can define a feasible schedule in this way.

The converse also holds. For a given feasible schedule, we assign U_i to *True* for all x_{ij} residing in TR, and U_i to *False* for all x_{ij} residing in FR. Clearly, by the two conditions of non-interfering literals stated above, each variable in U is uniquely set. Since each c_k is preceded by an literal x or $\overline{x}$ in TR (which means that the literal appears in C_k and it's value is *True* under the above assignment), F is satisfiable. □

We have deliberately used many distinct shifts to ease discussion. One could replace y and z by u, v_{ij} by x_{ij} and w_{ij} by $\overline{x_{ij}}$, thereby reducing J to $2P + m + 2 = 4m + 2$. The following colloraries are obvious:

Collorary 3.1. CSAP$(C|U|E)$ is NP-complete.

Collorary 3.2. CSAP$(N|U|S)$ is NP-complete.

Collorary 3.3. CSAP$(C|U|S)$ is NP-complete.

4 Polynomially Solvable Variants

We describe an algorithm to solve CSAP$(C|M|E)$ (and thus CSAP$(N|M|E)$) polynomially.

4.1 Algorithm

The algorithm is essentially a greedy algorithm which assigns slots in increasing order of shift numbers. As each workstretch is assigned from left to right, the following terms are defined. The *leftmost* of a workstretch at any point in the scheduling process is the slot where all left preceding slots have been assigned. The *rightmost* slot of a workstretch is literally the rightmost assignable slot. The *tail*

of a workstretch is the contiguous unassigned slots from the leftmost to the rightmost slots.

Due to the cyclic nature of the schedule, we regard the left and right positioning of slots not in terms of absolute indices, but considering wraparound.To simplify notation, $\sigma_{k,i-1}$ and $\sigma_{k,i+1}$ refer to the slots to the left and right of $\sigma_{k,i}$ respectively considering wraparound.

Let B be the set of day indices which have positive demands for shift j. Let A be the set of indices of workstretches whose leftmost slots' day indices are in B. A thus contains the potential workstretches for the assignment of shift j. For simplicity of notation, we will use σ as both the input and the output schedule. The algorithm is given as follows:

Input: $(I, J, K, D, \delta, \sigma)$, where δ is monotonic.
Output: a feasible schedule σ or **fail**.
Procedure:
1 **for** $j = 1$ **to** J **do**
2 **while** $(B \leftarrow \{i \leq I \mid D_{i,j} > 0\}) \neq \emptyset$ **do**
3 $A \leftarrow \{k \leq K \mid \sigma_{k,i}$ is leftmost slot **and** $i \in B\}$;
4 **if** $A = \emptyset$ **return fail**;
5 pick $k \in A$ s.t. k has the longest tail;
6 $\sigma_{k,i} \leftarrow j$; % $\sigma_{k,i}$ is leftmost slot of k
7 $D_{i,j} \leftarrow D_{i,j} - 1$;
8 **endwhile**
9 **endfor**

Line (5) picks a potential workstretch with the longest tail, ties broken arbitrarily. Note that even if a workstretch has a duplicating day index, there is no ambiguity of which index as far as Line (6) is concerned, since each workstretch has at most one leftmost slot.

4.2 Proof of Correctness

Theorem 4.1. The algorithm returns an instantiation of σ (i.e. a feasible schedule) if and only if there exists a feasible schedule.

Proof. If the algorithm exits successfully, then all slots in σ would have been assigned shifts in a monotonic fashion. Thus, σ is a feasible schedule. We prove the converse by showing that, if a feasible solution exists, then every assignment made in Line (6) augments a partial solution without violating feasibility. The following lemma, called the *Swapping Lemma*, is central to our proof.

Lemma 4.1. (*Swapping Lemma*) Let σ be a partial schedule in which shifts 1 to $j-1$ (for some j) have been assigned. Let σ^+ be σ plus one new assignment $\sigma_{k,i} \leftarrow j$, where $\sigma_{k,i}$ is a leftmost slot and k is the workstretch with the longest tail among all workstretches with leftmost slots. Then, σ^+ admits a feasible schedule if σ admits a feasible schedule.

Proof. Let σ^* represent a feasible schedule derived hypothetically from σ. If $\sigma^*_{k,i} = \sigma^+_{k,i} = j$, then σ^+ obviously admits a feasible schedule. Otherwise, let that particular shift j reside in $\sigma^*_{k',i}(k' \neq k)$ instead. Then, the following holds:

1. Since we schedule in non-decreasing shift numbers, $\sigma^*_{k,i} > \sigma^*_{k',i}$; and
2. by monotonicity, $\sigma^*_{k',i-1} \leq j$ and $\sigma^*_{k,i-1} \leq j$.

If $\sigma^+_{k',i}$ is also a leftmost slot, then the fact that our algorithm did not pick k' means that k is at least as long as k' in tail length. Otherwise, we claim that k' cannot possibly have a longer tail than k *from position* i as follows. Let k''s leftmost slot be i'. By monotonicity, slots $\sigma^*_{k',i'}$ to $\sigma^*_{k',i}$ have to be assigned j. If the claim is false, then k' would have been chosen instead of k since $D_{i',j} > 0$ at that point, a contradiction.

Thus, we have the scenerio as shown in Figure 3. We swap the content of $\sigma^*_{k,i}$ and $\sigma^*_{k',i}$. Surely, the content of $\sigma^*_{k',i}$ (i.e. j) may be moved to $\sigma^*_{k,i}$. The reverse is possible if $\sigma^*_{k',i+1} \geq \sigma^*_{k,i}$ or $\sigma^*_{k',i}$ is a rightmost slot; else, $\sigma^*_{k,i+1} > \sigma^*_{k',i+1}$, so we apply the Swapping Lemma recursively. In this way, k' will eventually hit the rightmost slot before k does because the latter is at least as long. Since the

swapping does not violate feasibility, we conclude that σ^+ also admits a feasible schedule. □

Now we return to Theorem 4.1. To prove that the algorithm incrementally builds a feasible solution, we reason by induction on the loop index j.

For $j = 0$, which is a blank schedule with only offday assignments, Theorem 4.1 obviously holds. Assume that after iteration $j-1$ ($j-1 < J$), the partial schedule admits a feasible schedule. In iteration j, a leftmost slot always exists by the induction hypothesis. By the Swapping Lemma, the assignment on Line (6) will admit a feasible schedule. □

4.3 Complexity Analysis

The matrix sum of D is the total number of working slots in σ, which is at most $K \cdot I$. Each **while** iteration decrements the matrix sum of D by 1 unit. Thus, the total number of iterations is $O(KI)$. B may be implemented as an $O(I)$-sized boolean vector, which is reset at the beginning of each **for** iteration and updated in constant time from one **while** iteration to the next (Line (2)). A may be implemented as an $O(K)$-sized boolean vector, which is written $O(KI)$ times (Line (3)) and read $O(KI)$ times (Line (5)). Hence, the worst-case time complexity is $O(\max(IJ, K^2I))$.

5 Conclusion

We presented a class of the manpower scheduling problems which is widely researched upon among the operations research community, and prove that majority of them are NP-hard. We showed that where shift changes are monotonic and demands are exact, a feasible schedule could be found in polynomial time for both the normal and cyclic schedules. One of our future research topics is to determine the complexity of the problem involving monotonic shift changes and slack demands, for both normal and cyclic schedules. Our contribution marks a step in formally analyzing the complexity of manpower scheduling problems. Other interesting research topics include finding computational complexities of and good algorithms for problems involving other scheduling constraints, in manpower scheduling as well as related types of resource scheduling.

Acknowledgments

I would like to thank Professor Osamu Watanabe for his valuable advice and suggestions.

References

[BK87] R. N. Burns and G. J. Koop. A modular approach to optimal multiple-shift manpower scheduling. *Operations Research*, Vol. 35, No. 1, pp. 100–110, 1987.

[BW90] N. Balakrishnan and R. T. Wong. A network model for rotating workforce scheduling problem. *Networks*, Vol. 20, pp. 25–32, 1990.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co, New York, 1979.

[GM86] F. Glover and C. McMillan. The general employee scheduling prolem: An integration of MS and AI. *Computers and Operations Research*, Vol. 13, No. 5, pp. 563–573, 1986.

[KJ91] M. M. Kostreva and K. S. Jennings. Nurse scheduling on a microcomputer. *Computers and Operations Research*, Vol. 18, No. 8, pp. 106–117, 1991.

[TK82] J. Tien and A. Kamiyama. On manpower scheduling algorithms. *SIAM Review*, Vol. 24, No. 3, pp. 275–287, 1982.

Shift\Day	1	2	3	4	5	6	7
1	1	1	0	1	1	1	0
2	0	1	2	2	0	0	2
3	2	2	1	0	3	2	1
4	1	0	1	1	1	2	1
5	1	2	0	0	0	1	1
6	1	1	3	2	1	0	0
7	0	0	1	1	2	0	1
8	2	1	1	1	0	1	0

(a)

Worker\Day	1	2	3	4	5	6	7
1	-	-	-	-	-	0	0
2	-	-	-	-	-	-	0
3	0	-	-	-	-	-	-
4	0	0	-	-	-	-	-
5	-	0	0	-	-	-	-
6	-	-	-	0	0	-	-
7	-	-	-	-	0	0	-
8	-	-	-	-	-	0	0
9	-	-	-	-	-	-	0
10	-	-	-	0	-	-	-

(b)

Worker\Day	1	2	3	4	5	6	7
1	4	5	6	7	7	0	0
2	1	2	3	4	4	5	0
3	0	1	2	2	3	4	5
4	0	0	2	2	3	4	7
5	8	0	0	1	3	3	4
6	6	6	7	0	0	3	3
7	5	5	6	8	0	0	2
8	3	3	6	6	7	0	0
9	3	3	4	6	6	8	0
10	8	8	8	0	1	1	2

(c)

Workstretch\Day	1	2	3	4	5	6	7
1	-	-	-	-	-	0	0
2	-	-	-	-	-	-	0
3	0	-	-	-	-	-	-
4	0	0	-	-	-	-	-
5	-	0	0	0	0	0	0
6	0	0	0	-	-	-	-
7	-	-	-	0	0	0	0
8	0	0	0	0	0	-	-
9	-	-	-	-	0	0	0
10	0	0	0	0	0	0	-
11	-	-	-	-	-	0	0
12	-	-	-	-	-	-	0
13	-	-	-	0	0	0	0
14	0	0	0	0	-	-	-

(d)

Workstretch\Day	1	2	3	4	5	6	7	1	2	3	4	5	6	7
1	-	-	-	-	-	-	0	0	0	0	0	0	0	0
2	0	-	-	-	-	-	-	0	0	0	0	0	0	0
3	0	0	-	-	-	-	-	-	0	0	0	0	0	0
4	0	0	0	-	-	-	-	-	-	-	0	0	0	0
5	0	0	0	0	0	-	-	-	-	-	-	0	0	0
6	0	0	0	0	0	0	-	-	-	-	-	-	0	0
7	-	-	-	-	-	-	0	0	0	0	0	0	0	0
8	-	-	-	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	-	-	-	-	-	-	-	-	0	0

(e)

Figure 1. In this example, I=7, J=8 and K=14. (a) shows a demand matrix; (b) shows an initial schedule pre-assigned with offdays; (c) shows a schedule assigned with shifts that satisfies the demand matrix; (d) shows the normal workstretch-based schedule derived from (b); and (e) shows the cyclic workstretch-based schedule derived from (b).

	1	2	3	4	5
1	u	v_{11}	x_{11}	c_1	-
2	u	v_{12}	x_{12}	c_2	-
3	u	v_{21}	$\bar{x}_{21}$	θ	-
4	u	v_{22}	$\bar{x}_{22}$	c_4	-
5	u	v_{31}	x_{31}	θ	-
6	u	v_{32}	x_{32}	c_3	-
7	-	w_{12}	$\bar{x}_{11}$	y	z
8	-	w_{11}	$\bar{x}_{12}$	y	z
9	-	w_{21}	x_{21}	y	z
10	-	w_{22}	x_{22}	y	z
11	-	w_{32}	$\bar{x}_{31}$	y	z
12	-	w_{31}	$\bar{x}_{32}$	y	z

Figure 2. Instance of CSAP constructed by reduction. ($\bar{x}ij$ represents the complement of x_{ij})

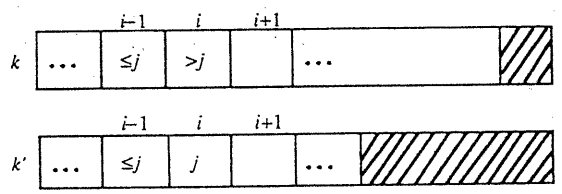

Figure 3. Contents of slots on rows k and k' in σ^*.