THE COLUMN-CIRCULAR, SUBSETS-SELECTION PROBLEM: COMPLEXITY AND SOLUTIONS

Fayez F. Boctor, Jacques Renaud

Centre de Service, d’Orientation et de Recherche sur la Compétitivité Internationale et l’Ingénierie de l’Entreprise Réseau (SORCIER)

Original manuscript : ISBN -
Version original : ISBN -
Série électronique mise à jour : 09-1998
One-line publication updated : 

Seria electrónica, puesta al día
THE COLUMN-CIRCULAR, SUBSETS-SELECTION PROBLEM: COMPLEXITY AND SOLUTIONS

Fayez F. Boctor\textsuperscript{1,2} and Jacques Renaud\textsuperscript{1,3}
\textsuperscript{1}Centre de Service, d’Orientation et de Recherche sur la Compétitivité Internationale et l’Ingénierie de l’Entreprise- Réseau, Université Laval,\textsuperscript{2}Faculté des Sciences de l’Administration, Université Laval, and\textsuperscript{3}Télé-Université, Université du Québec, Canada

Abstract

In this paper we study the complexity of a new class of a problem that we call the column-circular subsets-selection problem and we show that, under a special condition, it is a polynomially solvable problem. First we show that the column-circular set-partitioning, the column-circular set-covering, and the column-circular set-packing problems, among others, are special cases of the problem considered here. Then we present some of its applications. It is also shown that the optimal solution of some of the special cases of the column-circular subsets-selection problem can be obtained by solving a bounded number of totally-unimodular, linear-programming, sub-problems. In the case of column-circular set-partitioning, set-packing and set-packing with under-cover penalties problems, each of these linear sub-problems can be transformed into a shortest path problem. We provide some dynamic programming algorithms to solve the sub-problems of the column-circular subsets-selection problem and its special cases. Finally, a procedure to minimize the number of sub-problems to be solved is described.

Key words : Algorithms, Complexity, Networks, Scheduling, Set-covering, Set-packing, Set-partitioning.

1- Introduction

In this paper, a 0-1 vector is said to be a "consecutive-ones" or an "interval" vector if all its 1's occur consecutively and is said to be a "circular" vector if all its 0's are consecutive but its 1's are not (in this case it contains two blocks of consecutive 1's). A matrix is called a column- (respectively row-) interval matrix if we can arrange its rows (respectively columns) to show that all its columns (respectively rows) are interval or consecutive-ones vectors. Also, a matrix will be called column- (or row-) circular if, while we cannot show that it is an interval matrix, we can arrange its rows (columns), to show that it contains at least one circular column (respectively row) while the other columns (respectively rows) are interval vectors. Matrices which are both column-circular and row-circular are called twice-circular (see Figure 1).
Let us now present the class of problems we call the subsets-selection problem and the sub-
classes we call the interval and the column-circular subsets-selection problems. Let $M$ be a set of
$m$ elements, $N$ be a set of $n$ subsets of $M$, denoted $M_j$, $j=1, ..., n$, and $A=[a_{ij}: i=1, ..., m, j=1, ..., n]$ be a binary matrix where $a_{ij}$ takes the value 1 if and only if the element $i$ is included in (covered by) the subset $M_j$. The subsets-selection problem (SSP) is the problem of selecting the minimum cost collection of subsets $M_j$. Within this problem three cost elements are considered: $c_j$ the cost of selecting the subset $M_j$, $u_i$ the cost if the element $i$ is not covered by any selected subset (element $i$ is under-covered) and $v_i$ the cost of each over-covering of element $i$ (the number of times $i$ is over-covered equals the number of selected subsets that cover $i$, minus 1). Hereafter this problem is called the column-interval subsets-selection problem (CISSP) if $A$ is a column-interval matrix and is called the column-circular subsets-selection problem (CCSSP) if $A$ is a column-circular matrix.

In the following we present the SSP as a mathematical model and we show that the set-
partitioning problem (SPP), the set-covering problem (SCP) and the set-packing problem (SKP)
can be seen as special cases of the SSP. Let $c$ be an $n$ real-number vector, $u$ and $v$ be $m$ real-
number vectors, $x$ be $n$ binary vector, $y$ be $m$ binary vector, $z$ be an $m$ non-negative real-number
vector and $A$ be an $mn$ binary matrix, then the subsets-selection problem is the problem:

\[
\text{Min } \{ cx+uy+vz \mid x \text{ and } y \text{ binary, } z \geq 0, Ax+Iy-Iz=1 \}.
\]

Without loss of generality, we assume that $A$ does not contain neither identical columns nor
identical rows. Also notice that if both $u$ and $v$ are strictly positive, the optimal solution of (1) is
such that $y_i, z_i=0$ for all values of $i$. 

**Figure 1:** Some special matrix structures:
(a) column-interval matrix, (b) column-circular matrix, (c) twice-circular matrix.
It is easy to see that if the elements of both $u$ and $v$ are set to sufficiently large positive numbers then problem (1) will have the same optimal solution as the following problem known as the set-partitioning problem (SPP):

$$\text{Min } \{cx \mid x \text{ binary}, Ax=1\}, \quad (2)$$

assuming that the latter problem is feasible. Also, if the elements of $u$ are set to sufficiently large numbers and $v$ is set to zero then the optimal solution of problem (1) will be the same as that of the following problem known as the set-covering problem (SCP):

$$\text{Min } \{cx : x \text{ binary}, Ax\geq1\}. \quad (3)$$

In addition, if the elements of $v$ are set to sufficiently large numbers and $u$ is set to zero then the optimal solution of problem (1) will be the same as that of the following problem known as the set-packing problem (SKP):

$$\text{Min } \{cx : x \text{ binary}, Ax\leq1\}. \quad (4)$$

Notice that if any element of $c$, say $c_j$, is positive, then the corresponding variable $x_j$ should take the value 0 in the optimal solution. Consequently this variable can be dropped from the problem. Also, the set-packing problem is usually written as a maximization problem rather than a minimization problem. Let $d=-c$, then the SKP can be written:

$$\text{Max } \{dx : x \text{ binary}, Ax\leq1\}. \quad (4')$$

Other special cases of (1) can be defined. For example, if the elements of $u$ are set to sufficiently large numbers then the optimal solution of (1) will be the same as that of what we may call the set-covering problem with over-cover penalties (SCPP):

$$\text{Min } \{cx+vz \mid x \text{ binary}, z\geq0, A\cdot1z=1\}. \quad (5)$$

Notice that if $v$ is set to zero then the optimal solution of (5) will be the same as that of the set-covering problem (3). Thus the set-covering problem can be seen as a special case of the SCPP.

Finally, if the elements of $v$ are set to sufficiently large numbers then the optimal solution of (1) will be the same as that of what we may call the set-packing problem with under-cover penalties (SKPP):

$$\text{Min } \{cx+uy \mid x \text{ and } y \text{ binary}, Ax+Iy=1\}. \quad (6)$$

Notice that if $u$ is set to zero then the optimal solution of (6) will be the same as that of the set-packing problem (4). On the other hand, if the elements of $u$ are set to sufficiently large numbers then the optimal solution of (6) will be the same as that of the set-partitioning problem (2). Thus the set-packing and set-partitioning problems can be seen as special cases of the SKPP.
As mentioned above, the subsets-selection problem where $A$ is a column-interval matrix will be called the column-interval subsets-selection problem and if $A$ is a column-circular matrix it will be called the column-circular, subsets-selection problem. The interval and column-circular, set-partitioning, set-covering or set-packing problems can be defined in the same way. Figure 2 shows the relationship between the SSP and its special cases.

Without loss of generality, in this paper we assume that every column of the matrix $A$ contains at least one zero. In the case where there is a column, say $k$, which does not contain any zeros (thus $x_k=1$ and $x_j=0$, $\forall j \neq k$ is a solution to the CCSSP), we can exclude this column, solve the remaining problem and compare the cost $c_k$ to the optimal solution of the remaining problem to deduce the overall optimal solution.

In this paper we show that, under a special condition, the column-circular subsets-selection problem and two of its special cases (the column-circular set-covering problem CCSCP and the column-circular set-covering problem with over-cover penalties CCSCPP) belong to the class of
P-problems. Besides, we show that this condition is always valid for the CCSPP (column-circular set-partitioning problem), the CCSKP (column-circular set-packing problem) and the CCSKPP (column-circular set-packing problem with under-cover penalties) and consequently these three special cases of the CCSSP always belong to the class of P-problems. Also we provide some simple algorithms to solve these problems and study their complexity.

2- Some applications

The column-circular subsets-selection problems can be used to solve some scheduling and routing problems. Consider, for example, the vehicle routing problem where we want to select, among $n$ candidate routes (a route visits a subset of customers), the collection of routes which allows us to visit each customer, preferably once, while minimizing the sum of: (1) the cost of the selected routes, (2) the cost or penalty of non-visited customers, if any, and (3) the cost or penalty of customers visited more than once. Assume that the set of $m$ customers is ordered, that each of the candidate routes visits only a group of customers that are consecutive according to the defined order (we consider the first customer as consecutive to the last one, thus the set of candidates includes routes such the one that visits customers: $k,k+1,...,m-1,m,1,2,...,p$), and that there is no pair of consecutive customers which do not appear together on at least one candidate route. Then our route selection problem can be formulated as a CCSSP. Figure 3 shows an example of the described vehicle routing problem.

<table>
<thead>
<tr>
<th>Customer</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Overcover</td>
<td>44</td>
<td>44</td>
<td>40</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>60</td>
<td>44</td>
<td>44</td>
<td>28</td>
<td>63</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
</tr>
<tr>
<td>Penalty</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>9</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3: A numerical example

Foster and Ryan\cite{Foster} examined the special case of the above problem where each customer should be visited once and only once, and where there exists a pair of consecutive customers which do not appear together on any candidate route. In this case, the problem can be formulated as a set-partitioning problem and it is easy to rearrange the constraints matrix rows to show that all its
columns are interval vectors. In other words, this problem can be formulated as a column-interval set-partitioning problem (CISPP). To show that in this case the constraints matrix is a column-interval matrix, we can arrange its rows as follows. Let \( s \) and \( s+1 \) be the two customers which do not appear together on any candidate route (thus, there is no column that contains a 1 on both position \( s \) and position \( s+1 \)), then if we rearrange the rows in the order: \( s+1, s+2, \ldots, m, 1, 2, \ldots, s-1, s \), we can see that all columns are interval vectors. Since column-interval matrices are totally unimodular (see Gondran \(^2\) page 14 and Schrijver \(^3\) page 279), the column-interval set-partitioning problem belongs to the class of P-problems. This same matrix structure was first observed by Veinott and Wagner \(^4\) in a class of scheduling problems. Three other problems which exhibit the same structure were studied by Segal \(^5\) (the telephone-operator scheduling problem), by Shepardson and Marstern \(^6\) (the one-part duty crew scheduling), and by Darby-Dowman and Mitra \(^7\) (a problem they called the extended set-partitioning problem).

More recently, Ryan et al. \(^8\) indirectly discussed the more general case of the above-mentioned vehicle routing problem where there is no pair of customers which do not appear together on at least one of the candidate routes. They showed how to solve this problem by solving a number of set-partitioning problems with totally-unimodular constraints matrices. Separately, Boctor and Renaud \(^9\) identified the column-circular set-partitioning problem and showed that it belongs to the class of P-problems. Both Ryan et al. \(^8\) and Boctor and Renaud \(^9\) showed how to solve the problem by solving a bounded number of shortest path sub-problems. Hereafter, we generalize these previous results and we show that other problems, like the column-circular set-packing problem with and without under-cover penalties, belong to the class of P-problems. Even we show that, under a special condition, the column-circular subsets-selection problem as well as the column-circular set-covering problem with and without over-cover penalties are polynomially solvable.

Finally we mention that Bartholdi et al. \(^10\) studied the cyclic staffing problem with twice-circular matrices. There are two important differences between their problem and the one studied in this paper. First, the decision variables in staffing problems are integer (not binary) variables and, second, the matrix they considered is twice circular.

3- Problem Classification

The purpose of this section is to prove that, under a special condition, the column-circular subsets-selection problem and two of its special cases (the column-circular set-covering with and without over-cover penalties) can be solved by a polynomial algorithm. Also, we show that this condition is always valid for the column-circular set-partitioning and the column circular set-
packing with and without under-cover penalties. Consequently, these three special cases of the CCSSP are polynomially solvable problems and belong to the class of P-problems.

**Proposition 1**: The column-interval subsets-selection problem (CISSP):

\[
\min \{cx+uy+vz \mid x\text{ and } y\text{ binary, } z \geq 0, Ax+ly-Iz=1\},
\]

where \( A \) is a column-interval matrix

is a polynomially solvable problem.

**Proof**: Obviously the constraints matrix of this problem is totally unimodular as it is composed of three interval matrices (thus totally unimodular matrices, see Gondran and Schrijver): \( A \), which is a column-interval matrix, and two identity matrices \( I \). Thus, the column-interval subsets-selection problem can be solved in a polynomial time by any polynomial linear programming algorithm after dropping the binary constraint on \( x \) and \( y \).

**Corollary 1.1**: The column-interval set-packing problem (with and without under-cover penalties), the column-interval set-partitioning problem, and the column-interval set-covering problem (with and without over-cover penalties), belong to the class of P-problems.

**Proposition 2**: Under the condition that the optimal solution is such that there is at least two consecutive elements which are not simultaneously covered by any of the selected subsets, the column-circular subsets-selection problem (CCSSP):

\[
\min \{cx+uy+vz \mid x\text{ and } y\text{ binary, } z \geq 0, Ax+ly-Iz=1\},
\]

where \( A \) is a column-circular matrix with at least one zero in each column

is a polynomially solvable problem.

**Proof**: Under the stated condition, the CCSSP (column-circular subsets-selection problem) is a polynomially solvable problem as its optimal solution can be obtained by solving a bounded number of CISSPs (column-interval subsets-selection problems). In fact, as there is no column that covers all the \( m \) elements and as the optimal solution is such that at least two consecutive elements are not simultaneously covered by any of the selected subsets, the optimal solution can be obtained by solving \( m \) sub-problems. The first can be deduced from the CCSSP by discarding all the subsets (columns) that simultaneously covers both the first and the \( m^{th} \) elements (among others), the second sub-problem discards all the subsets that simultaneously covers both the second and first elements, and the last sub-problem discards all the subsets that simultaneously covers both elements \( m \) and \( m-1 \). As was indicated in the previous section it is easy to rearrange the matrix rows in order to show that each of these sub-problems is a column-interval subsets-selection problem, i.e., a polynomially solvable problem. Thus, under the stated condition, the
CCSSP is polynomially solvable as it can be solved by solving a bounded number of polynomially solvable sub-problems. □

It seems to us that this special condition should be observed in many practical cases. However, even if it is not, the above solution scheme should lead to a quite good or a near optimal solution.

**Proposition 3**: All the feasible solutions of the column-circular set-packing problem with under-cover penalties (CCSKPP):

\[
\min \{cx+uy \mid x \text{ and } y \text{ binary}, Ax+Iy=1\},
\]

where \( A \) is a column-circular matrix with at least one zero in each column (9), are such that at least two consecutive elements are not simultaneously covered by any of the subsets of the considered solution.

**Proof**: As there is at least one zero in each column, any feasible solution should contain two or more columns (two or more variables \( x \) and/or \( y \) should have the value 1). Then if every pair of consecutive elements are simultaneously covered, among others, by one of the solution columns, at least one element is covered by two or more columns. This implies that the solution is not feasible as the constraint corresponding to this element will be violated. □

**Corollary 3.1**: The column-circular set-packing problem with under-cover penalties (CCSKPP) and its special cases, the column-circular set-partitioning problem (CCSPP) and the column-circular set-packing problem (CCSKP), belong to the class of P-problems. This can be directly deduced from propositions 2 and 3 together.

### 4- Solution Techniques

In the proof of proposition 2 we suggested solving the column-circular subsets-selection problem (CCSSP) by solving \( m \) column-interval subsets-selection sub-problems. This same scheme can be used to solve any of its special cases and will lead to the optimal solution if the condition stated in proposition 2 is valid. Although, we can use either the Simplex method or any polynomial linear programming technique to solve these \( m \) totally unimodular sub-problems, in the following we provide a more efficient solution technique. Also we provide some efficient techniques to solve the column-interval set-partitioning problem (CISPP), the column-interval set-packing problem (CISKP) without and with under-cover penalties (CISKPP), the column-interval set-covering problem (CISCSP) without and with over-cover penalties (CISCPP), as well as the column-interval subsets-selection problem (CISSP). Again, without loss of generality, we assume that there is no subset that covers all the elements of \( M \) (thus every column of \( A \) contains at least one zero).
4.1 The set-partitioning problem

Let us first present a method to solve the column-interval set-partitioning problem (CISPP) which is the basic block to solve the column-circular set-partitioning problem (CCSPP). As we assume that each candidate subset covers only a group of consecutive elements and no subset covers all the \( m \) elements, it is easy to see that in the optimal solution there will be at least two consecutive elements which are not simultaneously covered by any one of the selected subsets. This implies that the optimal solution of the CCSPP can be obtained by solving a series of \( m \) column-interval set-partitioning sub-problems where the \( i^{th} \) sub-problem is the one where we discard all the candidate subsets which simultaneously cover both element \( i-1 \) and element \( i \). The 1st sub-problem is the one we get by discarding all the routes which simultaneously include both element 1 and element \( m \) (recall that we consider the first element as consecutive to the last one). Each of these totally-unimodular, linear-programming sub-problems can be transformed into a shortest path problem in an acyclic network (graph) and can be solved in a polynomial time by a dynamic programming algorithm.

To transform the \( i^{th} \) column-interval set-partitioning sub-problem (where there is no subset that simultaneously covers elements \( i-1 \) and \( i \)) into a shortest path problem we use an approach similar to that used by Shepardson and Marstern\(^6\). In summary, we construct a network with \( m+1 \) nodes (one for each element except element \( i \) which is represented by both nodes 1 and \( m+1 \)) and at most \( n-1 \) edges (one for each subset). If the \( j^{th} \) subset covers elements \( k,k+1,...,p \), then the associated edge runs from node \( k-i+1+m(\text{mod } m) \) to node \( p-i+2+m(\text{mod } m) \) with length \( c_j \). There is only one exception to this rule, edges associated with subsets where the last element is \( i-1 \) will end at node \( m+1 \). The column-interval set-partitioning sub-problem can then be solved by finding the shortest path between node 1 and node \( m+1 \) in the constructed network.

Back to the numerical example of Figure 3, the optimal solution of the embedded CCSPP (column-circular set-partitioning problem) can be obtained by solving \( m \) CISP (column-interval set-partitioning) sub-problems. The first sub-problem can be deduced by discarding all the routes that visits both customer 1 and customer 9. Thus, in this sub-problem we discard routes 14, 15, 16, 17 and 18. The resulting CISP sub-problem can be solved by finding the shortest route between node 1 and node 10 in the network shown in Figure 4 which was constructed following the rules given in the previous paragraph.

Note that this type of network does not contain any circuits since \( t<s \) for each edge \((t,s)\) (in Wagner\(^1\) page 180, this type of network is termed an acyclic network; hereafter we use the same term). Therefore, the shortest path length, noted \( F_{m+1} \), and the shortest path can be determined by
the following simple recursive algorithm, called *Algorithm 1* (this is a slightly modified version of the algorithm presented in Wagner\textsuperscript{11}, p.235):

$$
F_1 = 0, \\
F_s = \min \{ W, \min_{(t,s) \in E} \{ c_{ts} + F_t \} \}; \quad s=2,\ldots,m+1, \\
(10)
$$

where $F_s$ is the shortest path length from node 1 to node $s$, $E$ is the set of network edges, $c_{ts}$ is the length of edge $(t,s)$ and $W$ is a large number (e.g., $W > \sum_{(t,s) \in E} \{ c_{ts} \}$). Also note that this algorithm will identify situations where the sub-problem does not have a feasible solution (in this case, node 1 is not connected to node $m+1$). If there is no feasible solution to the sub-problem, the algorithm will give $F_{m+1} = W$.

**Figure 4**: The shortest path problem

Applying Algorithm 1 to the shortest path problem of Figure 4 we see that the shortest path between node 1 and node 10 is the path 1-4-6-10 which has a length of 209 units. Thus the optimal solution of the corresponding column-interval set-partitioning problem (which, in this case, is also the optimal solution of the whole CCSSP) is to use route 2 (visiting customers 1,2 and 3), route 7 (visiting customers 4 and 5), and route 13 (visiting customers 6,7,8 and 9). The calculations required by Algorithm 1 are summarized in Table 1 where the value given in the cell on line $t$ and column $s$ is $c_{ts} + F_t$. Notice that as node 2 is not linked to node 1, we obtained $F_2 = W$.

**Proposition 4**: The complexity of Algorithm 1 when applied to the CISPP is $O(n)$.

**Proof**: This is obvious as the total number of edges in the constructed network is at most $n-1$ and the number of basic operations in the algorithm is linearly proportional to the number of edges. $\square$
Corollary 4.1: The CCSPP can be solved to optimality by an algorithm of complexity $O(mn)$. This is easy to see as we solve the CCSPP by solving at most $m$ CISPP.

Table 1: Algorithm 1 applied to the network of Figure 4
(bold figures in the last line indicate the shortest path)

<table>
<thead>
<tr>
<th>From node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>To node $s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>W</td>
<td>52</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>W+92</td>
<td>W+99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>136</td>
<td>146</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>117</td>
<td>127</td>
<td>132</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>235</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>209</td>
<td>209</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$F_s$</td>
<td>W</td>
<td>52</td>
<td>65</td>
<td>136</td>
<td>117</td>
<td>127</td>
<td>132</td>
<td>209</td>
<td>209</td>
<td></td>
</tr>
<tr>
<td>from node</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

4.2. The set-packing problem

The column-interval set-packing problem (CISKP) without under-cover penalties can be solved by constructing the previously described network and searching within this network for a chain or a series of chains such that: (1) no arc appears more than once in the selected chain or chains, (2) no node is adjacent to more than two arcs belonging to the selected chain or chains, and (3) the sum of the involved arc lengths is minimal. This can be obtained by applying the following recursive algorithm, called Algorithm 2 (recall, since we write the set-packing problem as a minimization problem, all cost elements $c_{ts}$ are non positive. Otherwise the corresponding arc will not be includes in the shortest path):

$$F_1 = 0,$$
$$F_s = \min \{W, \min_{(t,s) \in E, \text{gst} \leq t} \{c_{ts} + F_g\}; \quad s = 2, 3, \ldots, m+1. \quad (11)$$

Although Algorithm 2 is quite simple, it is more efficient to deal with the CISKP as a column-interval set-packing problem with under-cover penalties (CISKPP) that are nil. To solve the CISKPP we use the following method. First, construct a network as above and add $m$ extra edges. The $i$th extra edge links node $i$ to node $i+1$ and its cost $c_{i,i+1}$ equals the penalty cost $u_i$ (notice that the augmented network is
also acyclic). Then use Algorithm 1 to determine the shortest path in the augmented network.

Let us go back again to the numerical example of Figure 3. To get the optimal solution of the embedded column-circular set-packing problem with the indicated under-cover penalties (CCSKPP), we have to solve \( m \) CISKPP (column-interval set-packing sub-problems with under-cover penalties). The first one is deduced by discarding every route visiting both customer 1 and customer 9; i.e., routes 14, 15, 16, 17 and 18.

Figure 5 gives the corresponding augmented network and Table 2 summarizes the calculations required when Algorithm 1 is applied to determine the shortest path between node 1 and node 10 in the augmented network. Here also each cell \((t,s)\) gives the value of \( c_{ts} + F_t \). This table shows that the shortest path in the augmented network is the path 1-4-6-10 which has a length of 209 units. Notice that this is the optimal solution of the first column-interval set-packing sub-problem but not the optimal solution of the whole CCSKPP (column-circular set-packing problem with under-cover penalties). The optimal solution of the CCSKPP is to use route 9 (visiting customers 4,5,6 and 7), route 18 (visiting customers 1,2,3 and 9) and not to visit customer 8. The total cost of this solution is 208.

![Figure 5: The augmented network](image)

**Proposition 5:** The complexity of Algorithm 1 when applied to the CISKPP is \( O(n+m) \).
\textbf{Proof}: This is obvious as the total number of edges in the augmented network is at most $n+m-1$ and the number of basic operations in the algorithm is linearly proportional to the number of edges. \□

\textbf{Corollary 5.1}: The CCSKPP can be solved to optimality by an algorithm of complexity $O(mn+m^2)$. This easy to see as we solve the CCSKPP by solving at most $m$ CISKPP.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
From node & \multicolumn{9}{c|}{To node $s$} \\
\hline
\textit{t} & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
1 & 44 & 52 & 65 &  &  &  &  &  &  \\
2 & 84 &  & 136 & 143 &  &  &  &  &  \\
3 &  & 96 & 136 & 146 &  &  &  &  &  \\
4 &  &  & 109 & 117 & 127 & 132 &  &  &  \\
5 &  &  &  & 169 & 190 & 235 &  &  &  \\
6 &  &  &  &  & 161 & 209 & 209 &  &  \\
7 &  &  &  &  &  &  &  &  & 155 \\
8 &  &  &  &  &  &  &  &  & 195 \\
9 &  &  &  &  &  &  &  &  & 239 \\
\hline
$F_s$ & 44 & 52 & 65 & 109 & 117 & 127 & 132 & 195 & 209 \\
\hline
\hline
\end{tabular}
\caption{Algorithm 1 applied to the network of Figure 5 \\
(bold figures in the last line indicate the shortest path)}
\end{table}

\subsection*{4.3- The set-covering problem}

The column-interval set-covering problem (CISCP) can be solved by constructing an acyclic network identical to the one used for the CISPP (column-interval set-partitioning problem) and searching within this network for a chain or a series of chains such that: (1) no arc appears more than once in the selected chain or chains, (2) each node is adjacent to at least one arc, and (3) the sum of the arc lengths involved is minimal. This can be obtained by applying the following recursive algorithm, called Algorithm 3:

\begin{alignat}{2}
F_1 &= 0, \\
F_s &= \min \{ W, \min_{(t,s) \in E, t \leq g < s} \{ c_{ts} + F_g \} \}; & \quad & s=2,\ldots,m+1,
\end{alignat}

(12)
Table 3 summarizes the calculations required by algorithm 3 when applied to the CISCP (with no over-cover penalties) deduced from the problem of Figure 3 by discarding every route visiting both customer 1 and 9. The corresponding network is given in Figure 4. Unlike Tables 1 and 2, in Table 3 the cell \((t,s)\) indicates (if and only if the edge \((t,s)\) \(\in E\)) the value of \(\min_{t \leq g < s} \{c_{ts} + F_g\}\) and not the value of \(c_{ts} + F_t\). The table indicates that the optimal solution is to use routes 2, 7 and 13 which do not lead to any over-covers. Notice however that the optimal solution for the whole problem without discarding any route is to use routes 2 (visiting customers 1, 2 and 3), route 9 (visiting customers 4, 5, 6 and 7) and route 15 (visiting customers 1, 8 and 9) which over-covers customer 1 with a total cost of 207.

Table 3: Algorithm 3 applied to the network of Figure 4
(bold figures in the last line indicate the optimal solution)

<table>
<thead>
<tr>
<th>From node</th>
<th>To node s</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t)</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>(W)</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>(F_s)</td>
<td>(W)</td>
</tr>
<tr>
<td>from node</td>
<td>1</td>
</tr>
</tbody>
</table>

Algorithm 3 can be extended to deal with the column-interval set-covering problem with over-cover penalties (CISCPP). The extended recursive algorithm, called Algorithm 4 is:

\[
F_1 = 0, \\
F_s = \min \left[ W, \min_{(t,s) \in E, t \leq g < s} \left\{ c_{ts} + F_{g} + \sum_{t \leq k < g} v_k \right\} \right]; \quad s=2,\ldots,m+1, \quad (13)
\]

**Proposition 6:** The complexity of Algorithm 4 when applied to the CISCP or to the CISCPP is \(O(mn)\).
Proof: The total number of edges in the constructed network is at most \( n-1 \) and for each edge the number of basic operations at most equals \( m \). Thus the complexity of Algorithm 4 is \( O(nm) \).

Corollary 6.1: Assuming the condition stated in proposition 2, the CCSCPP can be solved to optimality by an algorithm of complexity \( O(nm^2) \). This easy to see as we solve the CCSCPP by solving at most \( m \) CISCPP. Notice that if this condition is not valid, the obtained solution can be seen as an approximate (heuristic) solution.

4.4- The subsets-selection problem

Finally to solve the column-interval subsets-selection problem (CISSP) we may construct an augmented acyclic network identical to the one used to solve the column-interval set-packing problem with under-cover penalties and then apply Algorithm 4 to this network.

Proposition 7: The complexity of Algorithm 4 when applied to the CISSP is \( O(m^2+mn) \).

Proof: The total number of edges in the augmented network is at most \( m+n-1 \) and for each edge the number of basic operations at most equals \( m \). Thus the complexity of Algorithm 4 is \( O(m^2+mn) \).

Corollary 7.1: Assuming that the condition stated by proposition 2 is valid, the CCSSP can be solved to optimality by an algorithm of complexity \( O(m^3+m^2n) \). This easy to see as we solve the CCSPP by solving at most \( m \) CISSP. Notice that if this condition is not valid, the obtained solution can be seen as an approximate (heuristic) solution.

5- The Number of Sub-problems to be Solved

In the above it is suggested that the optimal solution of the column-circular subsets-selection problem can be obtained by solving \( m \) column-interval subsets-selection sub-problems. This is also valid for all its special cases; for example the optimal solution of the column-circular set-partitioning problem can be obtained by solving \( m \) column-interval set-partitioning sub-problems. Fortunately, it is often possible to solve less than \( m \) sub-problems.

Let \( S=\{k,k+1,\ldots,p-1,p\} \) be a set of consecutive elements which are not covered all together by any candidate subset. Then, we can see that the optimum solution will be such that at least two consecutive elements belonging to \( S \) are not covered by the same subset. Therefore, we can determine the optimum solution by only solving the \( m+p-k(\text{mod } m) \) sub-problems numbered
The problem now is to determine the set of consecutive elements which are not covered all together by any candidate subset and which have the smallest cardinal number; let such set be denoted \( S^* \). This can be done in two steps. First, we scan the set of candidate subsets to determine, for each value of \( i \), the subset with the largest cardinal number for which \( i \) is the first element; let this subset be denoted \( S_i \). The second step consists in determining, among the identified subsets \( \{S_i \mid i \in M\} \), the one (or any one of those) having the smallest cardinal number and which is not included in any other member of \( \{S_i \mid i \in M\} \). Let \( S_k = \{k, k+1, \ldots, p-1\} \) be the selected subset then we can use \( S^* = \{k, k+1, \ldots, p-1, p\} \) or \( S^* = \{k-1, k, k+1, \ldots, p-1\} \).

Applying this procedure to the example of Figure 3 we get

\[
S_1 = \{1, 2, 3\} \subset S_9, \quad S_2 = \{2, 3, 4, 5\},
S_3 = \{3, 4, 5\} \subset S_2, \quad S_4 = \{4, 5, 6, 7\}, \quad S_5 = \{5, 6, 7, 8\},
S_6 = \{6, 7, 8, 9\}, \quad S_7 = \{7, 8, 9, 1\}, \quad S_8 = \{8, 9, 1\} \subset S_7,
S_9 = \{9, 1, 2, 3\}.
\]

Thus we may choose \( S^* = \{2, 3, 4, 5, 6\} \) and to solve the corresponding CCSSP it is sufficient to solve four CISSP. Namely we have to solve the sub-problems numbered 3, 4, 5 and 6.

**Acknowledgement**

This work was partially supported by grants OPG 0036509 and OPG 0172633 from the Canadian Natural Sciences and Engineering Research Council (NSERC). The authors wish to thank the referees for their valuable comments and suggestions.

**References**


