A CYCLIC SELECTION ALGORITHM WITH SCHEDULING APPLICATIONS

Hesham K. Alfares[1], Shokri Z. Selim[2]

1: Associate professor, Systems Engineering Department, KFUPM
2: Professor, Systems Engineering Department, KFUPM

P.O. Box 5067, KFUPM, Dhahran 31261
E-Mail: hesham@ccse.kfupm.edu.sa

ABSTRACT

A new combinatorial algorithm is presented, which has potential applications in cyclic scheduling. Specifically, the question is: given n cyclic objects, how many cyclically distinct combinations of m ($m \leq n$) objects can be selected? In other words, if a subset m of these objects has one color, and the remaining ($n - m$) objects have another color, how many distinct arrangements can be made of all n cyclic objects? Integrating the tools of partition and permutation, an algorithm is developed for generating cyclically distinct selections. This algorithm includes a scheme for generating cyclically distinct permutations. A cyclic labor scheduling example is used to illustrate the applicability and usefulness of this algorithm.

Keywords: *Combinatorial Algorithms, Optimization, Cyclic scheduling, Cyclic Selection.*

الملخص

في هذا البحث يتم تطوير طريقة خوارزمية توافيقية يمكن أن تكون لها تطبيقات متعددة في الجدولة الدورية للقوى العاملة . ويمكن تحديد السؤال المطروح كما يلي : إذا كان لدينا عدد (ن) من الوحدات المرتبة دائريا بحيث أن الوحدة الأولى يمكن اعتبارها تالية للوحدة الأخيرة ، فما عدد المرات التي يمكن فيها اختيار (م) من هذه الوحدات (م < ن) بحيث يكون كل اختيار مختلفا عن الآخر دائريا ؟ ويمكن طرح السؤال بطريقة أخرى كالتالي : إذا كان لمجموعة (م) من هذه الوحدات لون معين وللوحدات (ن – م) الباقية لون آخر ، فكم عدد المرات التي يمكن فيها ترتيب جميع الوحدات ترتيبا مختلفا دائريا ؟ وبتجميع أداتي التقسيم والتبديل التوافيقيتين يتم تطوير أسلوب للحصول على تباديل مختلفة دائريا . وسوف تستخدم هذه الطريقة لحل مسألة جدولة دورية للقوى العاملة وذلك كمثال لتوضيح الفائدة العملية للطريقة المقترحة .

1. INTRODUCTION

Cyclic matrices are frequently encountered in optimization problems, such as network flow and labor scheduling problems. According to Montroll (1964), an *n*th-order matrix *A*, with

elements $a(j, k) \equiv a(k - j)$ that depend on only $k - j$, and that are periodic so that $a(k + n) = a(k)$, is called cyclic. The cyclic property of the constraint matrix can be exploited in order to develop efficient solutions of such problems. For example, Bartholdi et al. (1980) devise an efficient two solution procedures for labor days-off scheduling problems in which the matrix is row circular. Bartholdi et al. (1980) utilize the cyclic structure of the 0-1 matrix to solve the integer programming model parametrically as a series of bounded network flow problems.

Cyclic sequences and permutations are important problems in combinatorial theory. An example of a cyclic combinatorial problem is the Ménage problem. Krishnamurthy (1986, p. 92) states this problem as follows: how many ways can n married couples be seated at a circular table (with labeled seats) such that men and women alternate, with no husband sitting next to his wife? Another example is the necklace problem. The problem is stated by Krishnamurthy (1986, p. 101) as follows: how many distinct necklace patterns are possible with n beads, available in r different colors? The algorithm presented in this paper addresses a problem analogous to the necklace problem, which can be stated as follows: how many distinct necklace patterns are possible with n beads, m of which are of one color, and the rest are of another color?

The reader should not be deceived by the apparent simplicity or the limited applicability of the two above combinatorial problems. The men and women, or the two colors of the beads, should be thought of as 0s and 1s in a 0-1 cyclic matrix. The methodology to be presented in this paper is generally applicable to all integer linear programming (ILP) problems of the form: minimize $\mathbf{CX}$, subject to $\mathbf{AX} \geq \mathbf{b}$, where $\mathbf{X} \geq \mathbf{0}$ and integer, and the constraint coefficient matrix $\mathbf{A}$ is a 0-1 cyclic matrix. These ILP problems are known to be difficult to solve (NP hard) and have tremendous applications in many areas of optimization, such as cyclic scheduling and knapsack problems. A similar comment must also be stated for (5, 7) days-off scheduling example to be solved in Section 4. This example is purposefully chosen for its simplicity, to clearly illustrate the methodology. The proposed methodology is certainly not limited to this particular problem or to problems of similar size.

This paper uses concepts of combinatorial theory to develop a combinatorial algorithm for selecting cyclically distinct subsets out of a number of cyclic objects. Partition and permutation tools are combined, and a procedure for generating cyclic permutations is presented. The algorithm can be used to obtain optimum solutions of certain optimization problems with cyclic constraint matrices. Considering the columns (rows) of these matrices as cyclic objects, the algorithm can be used to select only cyclically distinct combinations of columns (rows). Using only a small subset of combinations to obtain optimum solutions, this process makes complete enumeration unnecessary. In comparison to complete enumeration, the reductions in the time and effort are tremendous.

Top

2. DESCRIPTION OF THE ALGORITHM

Given the total number of cyclic objects n, and the number of selected objects m ($m \le n$), the proposed combinatorial algorithm is designed to achieve two specific objectives. First, to determine the number of cyclically distinct combinations of m objects that can be selected. Second, to determine and describe each combination of the m selected objects. The two objectives are achieved mainly by combining two basic combinatorial tools: (i) partition of an integer, and (ii) permutation of objects. Therefore, these two tools will be briefly discussed before the steps of the algorithm are described.

Hall (1986, p. 31) defines the partition of a positive integer n into m parts ($m \le n$) as a representation of n as a sum of positive integers, expressed as

$$n = v_1 + v_2 + \ldots + v_m, \qquad v_i > 0, \qquad i = 1,\ldots, m \tag{1}$$

Wells (1971, p. 72) defines a permutation of n objects, as an order of the n objects. There are $n!$ permutations of n distinct objects. Unfortunately, there is no published procedure for generating cyclically distinct permutations, thus a new scheme will be developed for this purpose. Wells (1971, p. 70) defines a combination of n distinct objects taken m at time, an m-combination of n elements, as a selection of m of the n objects without regard to order. When the selection is made without replacement (i.e. when repetitions of the objects are not allowed), the number of m-combinations of n elements is $\binom{n}{m} = \frac{n!}{m!(n-m)!}$. When the objects are classified into m groups ($m \le n$), each group containing b_i similar objects, $i = 1,\ldots, m$, the number of different permutations becomes $\frac{n!}{b_1!b_2!\ldots b_m!}$, $\sum_i b_i = n$.

Our algorithm combines partition with permutation, and adds safeguards to eliminate cyclically redundant combinations. Effectively, a scheme for generating cyclically distinct permutations is applied to m partitions of an integer n. The algorithm proceeds in the following steps:

0. Given integers n and m ($m \le n$),

1. Partition n into m parts ($v_1, v_2, \ldots, v_m$) using a procedure that generates monotonic (non-decreasing or non-increasing) partitions. If $n \le 3$, or $n - m \le 1$, or $v_1 = v_m$ (all parts are equal), store the current combination and go to the next partition, otherwise, go to Step 2.

2. Fixing v_1, find the next permutation of the $m - 1$ remaining parts ($v_2, \ldots, v_m$).

   Cyclically compare the resulting (current) combination (including v_1) with all stored combinations of the current partition.

   (a) If there is no match, store the current combination and go to the next permutation,

   (b) otherwise, ignore the current combination and go to next permutation.

Top

The above description of the algorithm is purposely kept brief in order to avoid clouding the overall view of the algorithm with details. In order for the description to be complete, however, more detailed verbal description of steps 1 (partition) and 2 (permutation) will be given. In step 1, any monotonic partition procedure can be used. In our implementation of the algorithm, we used the procedure described by Lehmer (1964) to generate non-decreasing partitions ($1 \le v_1 \le v_2 \le \ldots \le v_m$). In step 2, we used Johnson's (1963) adjacent-mark procedure in our implementation of the algorithm. The algorithm represents each combination as a sequence of m partitions of n. Actually, these partitions are distances (differences) between successive numbers. The algorithm's representation can be converted to specific choices from the set (1, …, n). For example, the sequence of partitions (v_1, v_2, v_3) = (2, 5, 3) for n = 10 and m = 3 corresponds to selecting the numbers (2, 2 + 5, 2 + 5 + 3) = (2, 7, 10).

3. NUMBER OF CYCLICALLY DISTINCT SELECTIONS

The algorithm determines the number of cyclically distinct combinations of m objects that can be selected out of n cyclically arranged distinct objects. As stated earlier, the number of permutations of n distinct objects is equal to are n!. When one object is fixed to indicate cyclic permutation (e.g. arranging n people around a table), the number of permutations becomes $(n-1)$!. When clockwise or counterclockwise directions are considered equivalent, the number of permutations reduces to $(n-1)!/2$. As discussed above, for the linear (non-cyclic) case, the number of ways m elements can be selected out of n objects is given by the well-known formula $\binom{n}{m}$. Unfortunately, there is no general formula for the cyclic case, i.e., the number of cyclically distinct combinations of m objects that can be selected out of n cyclic objects.

Even though a general expression for the number of cyclic selections is not available, such an expression can be determined for three special cases. The first case is when $m \le 3$, and the second case is when $n - m \le 1$. In both cases, no permutations are required, and the number of cyclic selections is simply the number of partitions $P_m(n)$. The third special case applies to the remaining (unselected) $n - m$ objects. For the linear (non-cyclic) case, the number of combinations of the selected n items is equal to the number of combinations of the remaining $n - m$ items, or $\binom{n}{m} = \binom{n}{n-m}$. Obviously, this is also true for the cyclic case. Denoting the number of cyclically distinct combinations of m objects selected from n cyclic objects by $C_m(n)$, the three cases are summarized as follows:

$$C_m(n) = P_m(n), \qquad m \le 3, \text{ or } n - m \le 1 \tag{2}$$

$$C_{n-m}(n) = C_m(n) \tag{3}$$

For the remaining ranges of n and m, the values of $C_m(n)$ must be calculated by applying the steps of the cyclic selection algorithm. In order to calculate these values, and to facilitate the implementation of these steps, the algorithm was coded as a Microsoft PowerStation® FORTRAN program named CYCLE. The program generates all cyclic combinations $C_m(n)$ of m objects selected out of n cyclic objects.

4. AN APPLICATION EXAMPLE: THE (5, 7) CYCLIC LABOR SCHEDULING PROBLEM

The (5, 7) days-off scheduling problem is a practical and well-studied cyclic labor scheduling problem, which applies to organizations that operate seven days a week. The (5, 7) problem has a weekly schedule, in which each employee works five consecutive days and takes two consecutive days off. Allowing only consecutive pairs of off days, there are seven days-off patterns in each week. Given varying labor demands for each day of the week, the objective is to determine how many workers to assign to each days-off pattern in order to satisfy labor demands with the minimum workforce size. Vohra (1987) develops a formula for the minimum workforce size W. Mathematically, the problem is represented by the following integer programming model:

$$\text{Minimize } W = \sum_{j=1}^{7} x_j \tag{4}$$

$$\text{subject to} \quad \sum_{j=1}^{7} a_{ij} x_j \geq r_i, \qquad i = 1, 2, ..., 7 \tag{5}$$

$$x_j \geq 0 \text{ and integer}, \qquad j = 1, 2, ..., 7 \tag{6}$$

where

W = workforce size, i.e., total number of workers assigned

a_{ij} = 1 if day i is a workday for days-off pattern j, off on days j and j+1 mod 7, otherwise $a_{ij} = 0$

r_i = number of workers required on day i, $i = 1, 2, ..., 7$

x_j = number of workers assigned to weekly days-off pattern j,

The dual of the LP relaxation of the model, with dual variables y_i, $i = 1, 2, ..., 7$, is given by:

$$\text{maximize } W = \sum_{i=1}^{7} r_i y_i \tag{7}$$

subject to

$$\sum_{i=1}^{7} a_{ij} y_i \leq 1, \qquad j = 1, 2, ..., 7 \tag{8}$$

$$y_i \geq 0, \qquad i = 1, 2, ..., 7 \tag{9}$$

Using matrix notation, (5) can be represented as $AX \geq R$, while (8) can be represented as $A^{\mathrm{T}}Y \leq 1$, where A^{T} is the 7×7 dual constraint matrix shown below.

Top

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

To solve the dual problem we allocate the unit resource (right hand side of (8) which is equal to 1) among the dual variables (columns) in order to maximize the dual objective W. We may allocate the unit resource among any number m of selected columns, where $m = 1, ..., 7$. Since the seven columns (variables) of the dual constraint matrix A^T are cyclic, the number of cyclically-distinct dual solutions corresponds to the number of cyclically-distinct combination of m columns (variables) selected out of the seven columns of the matrix.

Running program CYCLE for $n = 7$ and $m = 1, ..., 7$, we obtain the 17 cyclic combinations shown in Table 1. The value of the workforce size W is obtained by multiplying the dual variables by the associated labor demands. The value of the dual variables corresponding to each combination is obtained by dividing the right-hand side of (8), equal to 1, over the maximum sum (among all rows) of variables in the selected m columns, which is given by

$$M = \max(\sum_{j \in SC} a_{ij}), \qquad i = 1, ..., 7 \tag{10}$$

The 17 dual solutions are shown in Table 1. The value in the last column (my_i) gives the ratio of the objective W with respect to the average labor demand for m days. For example, solutions 1-11, 13, and 16 all have $my_i = 1$, which means that the objective is equal to the average requirement for m days. Clearly, solution 1 dominates all these solutions, since if we choose $r_k = r_{max}$, then $W = r_{max}$ is is greater than or equal to the average labor requirement of any m days, where $m = 2 ,..., 7$. In solution 12, $my_i = 4/3$, thus the objective W is obtained by multiplying the average labor requirement of 4 days by 4/3. In solution 17, W is obtained by multiplying the average requirement of 7 days by 7/5. Although both solutions 14 and 15 have $my_i = 5/4 > 1$, they are both obviously dominated by solution 17.

Therefore, three dual solutions (numbered 1, 12, and 17 in Table 1) dominate all other solutions in terms of maximizing the objective W. Solution number 1 involves choosing one dual column (one non-zero dual variable) k, giving a workforce size W equal to r_k. Since the objective to maximize W, we choose the maximum r_k, $k = 1,..., 7$, or r_{max}. Solution number 12 corresponds to choosing four dual variables k, $k + 1$, $k + 3$, $k + 5$, yielding W equal to $(r_k + r_{k+1} + r_{k+3} + r_{k+5})/3$, $k = 1, ..., 7$. To maximize W, we choose the maximum $R_k = r_k + r_{k+1} + r_{k+3} + r_{k+5}$, $k = 1, 2, ..., 7$, or R_{max}. Solution number 17 involves choosing all seven dual variables, making W equal to $\sum r_k/5$.

Top

The minimum workforce size W is obtained by choosing the maximum value of the three solutions, and rounding up to integer values. The validity of rounding up dual continuous solutions to obtain optimum primal solutions has been established by several researchers such as Vohra (1987). The minimum workforce size W is given by:

$$W = \max \{ r_{\max}, \lceil \frac{R_{\max}}{3} \rceil, \lceil \frac{1}{5} \sum_{i=1}^{7} r_i \rceil \} \tag{11}$$

where

$$\lceil a \rceil = \text{smallest integer} \geq a$$

$$R_k = r_k + r_{(k+1) \bmod 7} + r_{(k+3) \bmod 7} + r_{(k+5) \bmod 7}, \quad i = 1, 2, ..., 7 \tag{12}$$

Table 1. Cyclically distinct combinations of m columns selected out of the seven dual matrix columns, and corresponding dual solutions for the (5, 7) problem

No.	m	SC	M	y_i	W	my_i
1*	1	K	1	1	r_k	1
2	2	$k, k+1$	2	1/2	$(r_k + r_{k+1})/2$	1
3	2	$k, k+2$	2	1/2	$(r_k + r_{k+2})/2$	1
4	2	$k, k+3$	2	1/2	$(r_k + r_{k+3})/2$	1
5	3	$k, k+1, k+2$	3	1/3	$(r_k + r_{k+1} + r_{k+2})/3$	1
6	3	$k, k+2, k+3$	3	1/3	$(r_k + r_{k+2} + r_{k+3})/3$	1
7	3	$k, k+3, k+4$	3	1/3	$(r_k + r_{k+3} + r_{k+4})/3$	1
8	3	$k, k+2, k+4$	3	1/3	$(r_k + r_{k+2} + r_{k+4})/3$	1
9	4	$k, k+1, k+2, k+3$	4	1/4	$(r_k + r_{k+1} + r_{k+2} + r_{k+3})/4$	1
10	4	$k, k+1, k+2, k+4$	4	1/4	$(r_k + r_{k+1} + r_{k+2} + r_{k+4})/4$	1
11	4	$k, k+1, k+3, k+4$	4	1/4	$(r_k + r_{k+1} + r_{k+3} + r_{k+4})/4$	1
12*	4	$k, k+1, k+3, k+5$	3	1/3	$(r_k + r_{k+1} + r_{k+3} + r_{k+5})/3$	4/3
13	5	$k, k+1, k+2, k+3, k+4$	5	1/5	$(r_k + r_{k+1} + r_{k+2} + r_{k+3} + r_{k+4})/5$	1
14	5	$k, k+1, k+2, k+3, k+5$	4	1/4	$(r_k + r_{k+1} + r_{k+2} + r_{k+3} + r_{k+5})/4$	5/4
15	5	$k, k+1, k+2, k+4, k+5$	4	1/4	$(r_k + r_{k+1} + r_{k+2} + r_{k+4} + r_{k+5})/4$	5/4
16	5	$k, k+1, k+2, k+3, k+4, k+5$	5	1/5	$(r_k + r_{k+1} + r_{k+2} + r_{k+3} + r_{k+4} + r_{k+5})/5$	1
17*	7	All 7 columns	5	1/5	$\sum_{k=1,..,7} r_k/5$	7/5

* = dominant solutions
m = number of selected columns
SC = selected columns, $k = 1, ...,7$, all values are mod 7
M = maximum number of non-zero coefficients in selected columns, defined by (19)
y_i = value of each selected (basic) dual variable: $y_i = 1/M$ if $i \in SC$, $= 0$ if $i \notin SC$
W = workforce size

Top

Using complete enumeration, Vohra (1987) obtains exactly the same formula specified by Equation (11) for the minimum workforce size *W*. Alfares (2000) also uses complete enumeration to determine the optimum solution of the (3, 7) cyclic days-off scheduling problem. The above example shows the effectiveness of the cyclic selection algorithm in exploiting the cyclic nature of constraint matrices to dramatically reduce computational effort. Instead of going through 7! = 5,040 linear combinations of the 7 dual variables, the optimum dual solution is determined by enumerating only 17 cyclic combinations.

5. CONCLUSIONS

A combinatorial cyclic selection algorithm, with applications in solving cyclic scheduling problems, has been presented. The process of cyclic selection determines all cyclically distinct combinations of *m* objects that can be selected out of *n* cyclic objects. The algorithm combines the two basic combinatorial tools of partition and permutation. A procedure for cyclic permutation is developed in order to enumerate all cyclically distinct combinations of the partitions of *n*. A FORTRAN program was coded and run in order to implement the algorithm and calculate the number of cyclic selections. A labor scheduling example is used to demonstrate the algorithm's applicability and effectiveness in reducing computational effort.

ACKNOWLEDGMENT

The authors would like to express gratitude to King Fahd University of Petroleum and Minerals for supporting this research effort.

REFERENCES

1. Alfares, H.K., 2000, "Dual-based optimization of cyclic three-day workweek scheduling," *Asia-Pacific Journal of Operational Research*, 17 (2), pp 137-148.
2. Bartholdi III, J.J., Orlin, J.B. and Ratliff, H.D., 1980, "Cyclic scheduling via integer programs with circular ones," *Operations Research*, 28 (5), pp 1074-1085.
3. Hall, M., Jr., 1986, *Combinatorial Theory*, Second Edition, John Wiley & Sons, New York.
4. Johnson, S.M., 1963, "Generation of permutations by adjacent transposition," *Mathematics of Computation*, 17, pp 282-285.
5. Krishnamurthy, V., 1986, *Combinatorics: Theory and Applications*, John Wiley & Sons, New York.
6. Lehmer, D.H., 1964, "The machine tools of combinatorics," in Beckenbach, E.F. (ed), *Applied Combinatorial Mathematics*, John Wiley & Sons, New York, pp 5-27.
7. Montroll, E.W., 1964, "Lattice statistics," in Beckenbach, E.F. (ed), *Applied Combinatorial Mathematics*, John Wiley & Sons, New York, pp 96-134.
8. Vohra, R.V., 1987, "The cost of consecutivity in the (5, 7) cyclic staffing problem," *IIE Transactions*, 19 (3), pp 296-299.
9. Wells, M.B., 1971, *Elements of Combinatorial Computing*, Pergamon Press, Oxford.