The paper presents a new method for solving the 0–1 linear programming problems (LPs). The general 0–1 LPs are believed to be NP-hard and a consistent, efficient general-purpose algorithm for these models has not been found so far. Cutting planes and branch and bound approaches were the earliest exact methods for the 0–1 LP. Unfortunately, these methods on their own failed to solve the 0–1 LP model consistently and efficiently. The hybrids that are a combination of heuristics, cuts, branch and bound and pricing have been used successfully for some 0–1 models. The main challenge with these hybrids is that these hybrids cannot completely eliminate the threat of combinatorial explosion for very large practical 0–1 LPs. In this paper, a technique to reduce the complexity of 0–1 LPs is proposed. The given problem is used to generate a simpler version of the problem, which is then solved in stages in such a way that the solution obtained is tested for feasibility and improved at every stage until an optimal solution is found. The new problem generated has a coefficient matrix of 0 s and 1 s only. From this study, it can be concluded that for every 0–1 LP with a feasible optimal solution, there exists another 0–1 LP (called a double in this paper) with exactly the same optimal solution but different constraints. The constraints of the double are made up of only 0 s and 1 s. It is not easy to determine this double 0–1 LP by mere inspection but can be obtained in stages as given in the numerical illustration presented in this paper. The 0–1 integer programming models have applications in so many areas of business. These include large economic/financial models, marketing strategy models, production scheduling and labor force planning models, computer design and networking models, military operations, agriculture, wild fire fighting, vehicle routing and health care and medical models

Keywords: 0–1 LP, unimodular, clique inequalities, feasible test, variable sum, double

UDC 519
DOI: 10.15587/1729-4061.2020.211793

DEVELOPMENT OF AN EXACT METHOD FOR ZERO-ONE LINEAR PROGRAMMING MODEL

Elias Munapo
PhD, Professor of Operations Research
Department of Statistics
and Operations Research
School of Economics
and Decision Sciences
North West University
Mmabatho Unit 5, Mahikeng, 2790,
Mafikeng, South Africa
E-mail: emunapo@gmail.com

Received date 05.09.2020
Accepted date 07.10.2020
Published date 30.10.2020

1. Introduction

Integer programming in general has very important applications in business. Research on the general integer problem has been going on for over 70 years [1, 2]. A consistent, efficient general-purpose algorithm for the integer has not been found so far. The 0–1 model has been used successfully to solve business problems for the past 70 years. With these models, the variables assume only binary variables. There are so many exact methods that were proposed for the 0–1 LPs. These include branch and bound [3], use of cuts [4–7], pricing [8, 9], heuristics and hybrids of these stated methods [10–12]. The major weakness of these methods is that they cannot completely eliminate the threat of combinatorial explosion for very large practical 0–1 problems.

In this paper, a technique to reduce the complexity of 0–1 LPs is proposed. The given problem is used to generate a simpler version of the problem, which is then solved in stages in such a way that the solution obtained is tested for feasibility and improved at every stage until an optimal solution is found. The new problem, which is a simpler problem is generated (in this case called a double) and has a coefficient matrix of 0 s and 1 s only.

The 0–1 integer programming models have applications in so many areas of business. These include large economic/financial models, marketing strategy models, production scheduling and labor force planning models, computer design and networking models, military operations, agriculture, fighting fire protection, vehicle routing and health care and medical models [13, 14]. Therefore, there is a need to develop an efficient general-purpose method for the 0–1 linear programming model.

2. Literature review and problem statement

The 0–1 LP is used in maximizing yield in Codon optimization. In the process, particular nucleotide bases sequences should be avoided or included [15]. Because of the two options (include or exclude), this problem can be formulated as a 0–1 LP. Currently, there is no efficient general-purpose algorithm for the formulated 0-1 LP for this problem. With such an important application in real life there is a need for an efficient general purpose algorithm for 0–1 LPs.

The paper [16] is aimed at finding better algorithms for solving parameter reduction problems of soft sets and gives their potential applications. In that paper, 0–1 formulations were used and from the comparison with the algorithm proposed by Ma et al. [16], experimental results showed that the proposed 0–1 formulation method for normal parameter

reduction was more efficient. Even though 0–1 formulations showed better results, there is no efficient general-purpose algorithm for these 0–1 models.

The paper [17] is on a general algorithm for converting the 0–1 integer linear programming problem into an optimal transition firing sequence problem (OFSP) of a Petri net (PN). The algorithm proposed in this paper was applied to a traveling salesman problem, a vehicle routing problem and an automated guided vehicles (AGV) routing problem. Cuts were derived using reachability analysis of the converted PN model. Computational results showed that the proposed algorithm using the reachability analysis was more efficient. The proposed algorithm [17] is not for all 0–1 linear integer models.

In the paper [18], a binary integer linear programming formulation of the rectangle blanket problem is presented. Four methods for solving the formulated problem were used, which were branch-and-price algorithm, constrained simulated annealing heuristic and two other heuristics. The fact that heuristics are being used shows that efficient exact approaches for 0–1 LPs are still not available.

In the paper [19], a new formulation based on the definition of new binary variables has been proposed to convert the grid-connected photovoltaic (GCPV) problem to the binary linear programming (BLP). The proposed method was able to find the global optimum solution faster than the evolutionary algorithms (EAs). This clearly shows how important the 0–1 LP is in solar systems.

The paper [20] shows the application of 0–1 integer programming in combination and placement of sustainable drainage system (SuDS) devices. It was shown that with 0–1 LPs, it helps to objectively choose the best SuDS device combination and placement scheme for cost-effective implementation.

Branch and bound related exact approaches that are normally used for the 0–1 LPs have the weakness that the number of sub-problems increases exponentially with an increase in the number of variables or size of the 0–1 LP problem [3]. There is definitely a need to improve solution methods for these 0–1 LPs.

3. The aim and objectives of the study

The aim of the study is to develop a method for the 0–1 LP model. To achieve the set aim, the following objectives have been set:

– to create a simpler problem in stages, of the given 0–1 LP with a coefficient matrix of 0 s and 1 s only;

– to illustrate the proposed algorithm by an example.

4. General 0-1 LP

Suppose the 0–1 LP is presented as given in (1).

Maximize $c_1x_1+c_2x_2+...+c_nx_n$.

Such that:

$$a_{11}x_1+a_{12}x_n+...+a_{1n}x_n\le b_1,$$

$$a_{21}x_1+a_{22}x_n+...+a_{2n}x_n\le b_2,$$

$$a_{m1}x_1+a_{m2}x_n+...+a_{mn}x_n\le b_m, \quad (1)$$

where $c_j\ge 0$, a_{ij} and b_j are constants, x_j is a binary variable, $i=1,2,...,m$ and $j=1,2,...,n$.

4. 1. Special relationship of 0–1 variables

A special relationship of binary variables is given in (2).

$$x_j+s_j=1, \quad (2)$$

where s_j is a slack variable and when $x_j=1$ then $s_j=0$ and vice versa. With this relationship, any maximization 0–1 LP can be converted into a minimization one and vice versa.

4. 2. Cutting plane & Branch and Bound approaches

Cutting plane and branch and bound approaches where the earliest exact methods for the 0–1 LP. Unfortunately, these methods on their own failed to solve the 0–1 LP model consistently and efficiently. The hybrids such as branch and cut, branch and price or branch cut and price were developed. The challenge with these hybrids is that they cannot completely eliminate the threat of combinatorial explosion for very large 0–1 LPs.

4. 3. Theorem

Matrix A is totally unimodular if it satisfies the following five conditions:

a) all entries of A are 0, 1 or –1;

b) the rows of A can be partitioned into two disjoint sets R_1, and R_2;

c) every column of A contains at most two nonzero entries;

d) if any column of A contains two nonzero entries of the same sign, then one is in a row of R_1 and the other in a row of R_2;

f) if any column of A contains two nonzero entries of opposite sign, then they are both in rows of R_1 or both in rows of R_2.

The proof and more on this theorem can be found in [20].

Unfortunately, the coefficient matrix of the general 0–1 LP is not totally unimodular. In this paper, constraints with 0, 1 and or –1 only as coefficients are created at every iteration. These special constraints are added to the current new problem and the process is repeated until a feasible solution is obtained.

5. Creating the new problem

5. 1. Initial new problem

From the problem given in (1), the new initial problem is created as given in (3).

Maximize $c_1x_1+c_2x_2+...+c_nx_n$.

Such that:

$$x_1+x_n+...+x_n\le\alpha_0, \quad (3)$$

where $\alpha_0\ge 0$ and integer.

The variable sum limit (α_0) is given by (4).

Maximize $\alpha_0=x_1+x_2+...+x_n$.

Such that:

$$a_{11}x_1+a_{12}x_n+...+a_{1n}x_n\le b_1,$$

$$a_{21}x_1+a_{22}x_n+...+a_{2n}x_n\le b_2,$$

$$a_{m1}x_1+a_{m2}x_n+...+a_{mn}x_n\le b_m. \quad (4)$$

The variable sum limit can also be accurately approximated using pre-processing. This significantly reduces the computational effort. Solving (3), we obtain the optimal binary solution as given in (5).

$$\left(Z^0,x_1^0,x_2^0,...,x_n^0\right). \quad (5)$$

The binary solution given in (5) satisfies the problem given in (3) and is optimal for that problem.

5. 2. First new problem

The initial solution is tested for feasibility and if feasible then it is optimal. Since we want all coefficients to be 0 and 1 only, the initial infeasible solution is used to generate clique inequalities. This is done by using the initial feasible solution to determine the original constraints that are violated and then these are used to generate clique inequalities. Suppose the initial optimal solution is given by (6) and the violated original constraint is given by (7).

$$(26,1,1,0,0,1,0). \quad (6)$$

$$3x_1+8x_2-5x_3+11x_4+9x_5+2x_6\le 10. \quad (7)$$

Using the initial optimal solution and the violated solution, we can generate the clique inequality given in (8).

$$x_1+x_2+x_5\le 1. \quad (8)$$

The clique inequalities generated from all the violated constraints are added to the initial infeasible problem and then solved to get the first optimal solution given in (9).

$$\left(Z^1,x_1^1,x_2^1,...,x_n^1\right). \quad (9)$$

The first optimal solution given in (9) is used to determine violated constraints. Once the violated constraints are known the process is repeated.

5. 3. Optimal solution

The optimal solution is obtained when the k^{th} optimal solution given in (10) is feasible.

$$\left(Z^k,x_1^k,x_2^k,...,x_n^k\right). \quad (10)$$

The solution in (9) is optimal in the sense that all constraints are not violated and is the largest in terms of the objective for (1).

6. Proposed algorithm for solving the 0–1 LP

The steps for the proposed algorithm are summarized as follows.

Step 1. Determine the variable sum limit (a_0) and use it to construct the initial new problem.

Step 2. Solve the problem to obtain an optimal integer solution $\left(x_1^k,x_2^k,...,x_n^k\right)$. Test this optimal solution for feasibility. If feasible then it is optimal to the original 0–1 LP else go to Step 3.

Step 3. Determine the original constraints that are violated by the infeasible solution. Use the original violated constraints to generate clique inequalities, add them to the current new problem. Return to Step 2. In diagram form, the proposed algorithm is presented in Fig. 1.

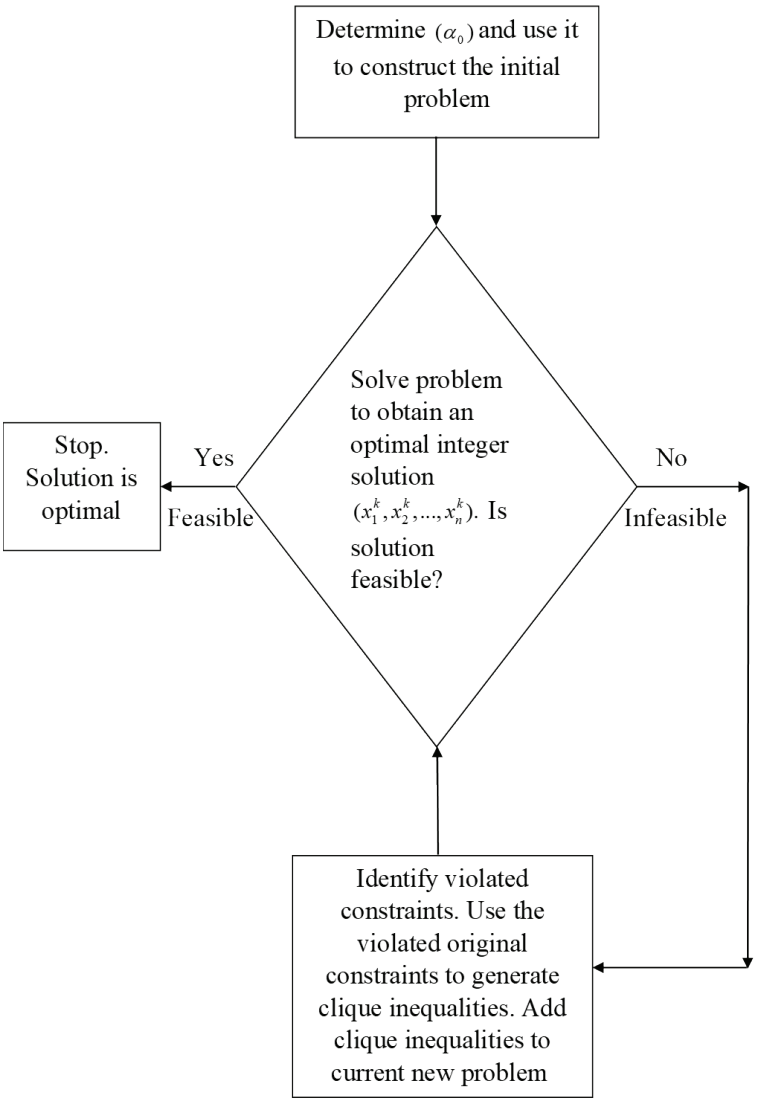

Fig. 1. Flow chart for the proposed method for 0–1 LPs

In every iteration, the problem is made easier to solve so as to get a new solution. The chance of the numbers of sub-problems exploding to unmanageable levels is put to a minimal.

6. 1. Numerical illustration

Maximize $8x_1+15x_2+6x_3+9x_4+7x_5$. Such that:

$$11x_1+13x_2+5x_3+7x_4+14x_5\le 29,$$

$$16x_1-12x_2+36x_3+11x_4+21x_5\le 35,$$

$$-13x_1+9x_2+13x_3+15x_4-5x_5\ge 15, \quad (11)$$

where $x_1,x_2,x_3,x_4,x_5\ge 0$ are binary variables. In the worst case, it takes 9 sub-problems to verify optimality.

Variable sum limit (a_0).

Maximize $\alpha_0=x_1+x_2+x_3+x_4+x_5$. Such that:

$$11x_1+13x_2+5x_3+7x_4+14x_5\le 29,$$

$$16x_1-12x_2+36x_3+11x_4+21x_5\le 35,$$

$$-13x_1+9x_2+13x_3+15x_4-5x_5\ge 15. \quad (12)$$

Solving we obtain (13).

$$x_1 + x_2 + x_3 + x_4 + x_5 \le 3.13,$$

i. e.

$$x_1 + x_2 + x_3 + x_4 + x_5 \le 3. \quad (13)$$

Since these are binary variables, we have (14).

Initial Problem and initial solution.

Maximize $8x_1 + 15x_2 + 6x_3 + 9x_4 + 7x_5$.

Such that:

$$x_1 + x_2 + x_3 + x_4 + x_5 \le 3, \quad (14)$$

where variables are binary.

Solving this we have the initial solution as given in (15):

$$(32,1,1,0,1,0). \quad (15)$$

Violations caused by initial solution.

Constraint 1: is violated and the clique $x_1 + x_2 + x_4 \le 2$, is generated.

Constraint 2: is satisfied.

Constraint 3: is violated and the clique $x_1 + x_2 + x_4 \le 2$, is generated.

Note that the two cliques are exactly the same therefore we take only one.

First Problem and First Solution.

We now add the generated clique constraint to get the First Problem as given in (16).

Maximize $8x_1 + 15x_2 + 6x_3 + 9x_4 + 7x_5$.

Such that:

$$x_1 + x_2 + x_3 + x_4 + x_5 \le 3,$$

$$x_1 + x_2 + x_4 \le 2, \quad (16)$$

where variables are binary.

Solving this we have the initial solution as given in (17):

$$(31,0,1,0,1,1). \quad (17)$$

Violations caused by First Solution.

Constraint 1: is violated and the clique $x_2 + x_4 + x_5 \le 2$, is generated.

Constraint 2: is satisfied.

Constraint 3: is satisfied.

Second Problem and Second solution.

We now add the generated clique constraints to get the Second Problem as given in (18).

Maximize $8x_1 + 15x_2 + 6x_3 + 9x_4 + 7x_5$.

Such that:

$$x_1 + x_2 + x_3 + x_4 + x_5 \le 3,$$

$$x_1 + x_2 + x_4 \le 2,$$

$$x_2 + x_4 + x_5 \le 2, \quad (18)$$

where variables are binary.

Solving this we have the initial solution as given in (19):

$$(30,1,1,0,0,1). \quad (19)$$

Violations caused by initial solution.

Constraint 1: is violated and the clique $x_1 + x_2 + x_5 \le 2$, is generated.

Constraint 2: is satisfied.

Constraint 3: is violated and the clique $x_3 + x_4 \ge 1$, is generated.

Second Problem and Second solution.

We now add the generated clique constraints to get the Second Problem as given in (20).

Maximize $8x_1 + 15x_2 + 6x_3 + 9x_4 + 7x_5$.

Such that:

$$x_1 + x_2 + x_3 + x_4 + x_5 \le 3,$$

$$x_1 + x_2 + x_4 \le 2,$$

$$x_2 + x_4 + x_5 \le 2,$$

$$x_1 + x_2 + x_5 \le 2,$$

$$x_3 + x_4 \ge 1, \quad (20)$$

where variables are binary.

Solving this we have the initial solution as given in (21):

$$(30,0,1,1,1,0). \quad (21)$$

In other words, the double for 0–1 LP for (12) is (20). In this paper, a double 0–1 LP is another 0–1 LP that gives exactly the same optimal solution as the original 0–1 LP. In addition, the coefficient matrix is made up of 0 s and 1 s only. It takes only 3 sub-problems to verify optimality.

Violations caused by initial solution.

Constraint 1: is satisfied.

Constraint 2: is satisfied.

Constraint 3: is satisfied.

There are no more violations thus the solution $(30,0,1,1,1,0)$ is optimal.

The coefficient matrix is given in (22):

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}. \quad (22)$$

7. Discussion of numerical illustration

We need a total of 3 iterations to solve the 0–1 LP given in the illustration. The total or final number of clique inequalities required to solve the problem is 4. From the numerical illustration, it can be noted that for every 0–1 LP there exists another 0–1 LP with exactly the same optimal solution but different constraints. It is not easy to determine this double 0–1 LP by inspection from the start but it can be obtained in stages as given in the numerical illustration. It is noted that the total number of sub-problems required to verify optimality is decreased from 9 to 3. An obvious disadvantage is that the number of clique constraints generated and added to the 0–1 LP increases from one iteration to the next.

The double obtained for the illustration is given in (20) and it is easier to solve than the original problem given in (3). The number of sub-problems required to verify optimality is reduced from 9 to 3. The coefficient matrix of the double obtained for the matrix is made up of 0 s and 1 s only as given in (22). If the coefficient matrix is made up of 0 s and 1 s, the closer it is to become unimodular. The closer a coefficient matrix is to become unimodular, the easier it is to solve by branch and bound related approaches. The properties of a totally unimodular matrix are presented in Section 4. 1.

Even though the number of cliques generated increases, what is pleasing is that these cliques significantly reduce the total number of iterations necessary to obtain the optimal solution.

A limitation of the proposed algorithm is that there are no computational results to compare with other methods.

8. Conclusions

1. A simpler problem of the original 0–1 LP was created in stages and its coefficient matrix was made up of 0 s and 1 s only. The 0–1 LPs made up 0 s and 1 s only are easier to solve than the general 0–1 LP.

2. The proposed method presented in this paper is illustrated in Section 5. 1. It can be noted that the variable sum limit can be accurately estimated, which will significantly reduce the number of computations.

Acknowledgments

We are grateful to the editor and the anonymous referees. We also take this opportunity to thank the NWU Faculty of Economic and Management Sciences (FEMS) for the unwavering research support.

References

1. Land, A. H., Doig, A. G. (1960). An Automatic Method of Solving Discrete Programming Problems. Econometrica, 28 (3), 497. doi: https://doi.org/10.2307/1910129
2. Dakin, R. J. (1965). A tree-search algorithm for mixed integer programming problems. The Computer Journal, 8 (3), 250–255. doi: https://doi.org/10.1093/comjnl/8.3.250
3. Al-Rabeeah, M., Munapo, E., Al-Hasani, A., Kumar, S., Eberhard, A. (2019). Computational Enhancement in the Application of the Branch and Bound Method for Linear Integer Programs and Related Models. International Journal of Mathematical, Engineering and Management Sciences, 4 (5), 1140–1153. doi: https://doi.org/10.33889/ijmems.2019.4.5-090
4. Djeumou Fomeni, F., Kaparis, K., Letchford, A. N. (2020). A cut-and-branch algorithm for the Quadratic Knapsack Problem. Discrete Optimization, 100579. doi: https://doi.org/10.1016/j.disopt.2020.100579
5. Brunetta, L., Conforti, M., Rinaldi, G. (1997). A branch-and-cut algorithm for the equicut problem. Mathematical Programming, 78 (2), 243–263. doi: https://doi.org/10.1007/bf02614373
6. Mitchell, J. E. (2001). Branch and cut algorithms for integer programming. Encyclopedia of Optimization. Kluwer Academic Publishers.
7. Padberg, M., Rinaldi, G. (1991). A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. SIAM Review, 33 (1), 60–100. doi: https://doi.org/10.1137/1033004
8. Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., Vance, P. H. (1998). Branch-and-Price: Column Generation for Solving Huge Integer Programs. Operations Research, 46 (3), 316–329. doi: https://doi.org/10.1287/opre.46.3.316
9. Savelsbergh, M. (1997). A Branch-and-Price Algorithm for the Generalized Assignment Problem. Operations Research, 45 (6), 831–841. doi: https://doi.org/10.1287/opre.45.6.831
10. Barnhart, C., Hane, C. A., Vance, P. H. (2000). Using Branch-and-Price-and-Cut to Solve Origin-Destination Integer Multicommodity Flow Problems. Operations Research, 48 (2), 318–326. doi: https://doi.org/10.1287/opre.48.2.318.12378
11. Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M. P. de, Reis, M., Uchoa, E., Werneck, R. F. (2005). Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. Mathematical Programming, 106 (3), 491–511. doi: https://doi.org/10.1007/s10107-005-0644-x
12. Ladányi, L., Ralphs, T. K., Trotter, L. E. (2001). Branch, Cut, and Price: Sequential and Parallel. Computational Combinatorial Optimization, 223–260. doi: https://doi.org/10.1007/3-540-45586-8_6
13. Taha, H. A. (2017). Operations Research: An Introduction. Pearson Educators.
14. Winston, W. L. (2004). Operations Research Applications and Algorithms. Duxbury Press.
15. Arbib, C., Pınar, M. Ç., Rossi, F., Tessitore, A. (2020). Codon optimization by 0-1 linear programming. Computers & Operations Research, 119, 104932. doi: https://doi.org/10.1016/j.cor.2020.104932
16. Han, B., Li, Y., Geng, S. (2017). 0–1 Linear programming methods for optimal normal and pseudo parameter reductions of soft sets. Applied Soft Computing, 54, 467–484. doi: https://doi.org/10.1016/j.asoc.2016.08.052
17. Kodama, A., Nishi, T. (2017). Petri net representation and reachability analysis of 0–1 integer linear programming problems. Information Sciences, 400-401, 157–172. doi: https://doi.org/10.1016/j.ins.2017.03.014
18. Demiröz, B. E., Altınel, İ. K., Akarun, L. (2019). Rectangle blanket problem: Binary integer linear programming formulation and solution algorithms. European Journal of Operational Research, 277 (1), 62–83. doi: https://doi.org/10.1016/j.ejor.2019.02.004
19. Bakhshi-Jafarabadi, R., Sadeh, J., Soheili, A. (2019). Global optimum economic designing of grid-connected photovoltaic systems with multiple inverters using binary linear programming. Solar Energy, 183, 842–850. doi: https://doi.org/10.1016/j.solener.2019.03.019
20. Commoner, F. G. (1973). A sufficient condition for a matrix to be totally unimodular. Networks, 3 (4), 351–365. doi: https://doi.org/10.1002/net.3230030406