I.J. Information Technology and Computer Science, **2015, 07, 28-34**
Published Online June 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2015.07.04

Great Deluge Algorithm for the Linear Ordering Problem: The Case of Tanzanian Input-Output Table

Amos Mathias
Department of Science Mathematics and Technology Education, University of Dodoma, Box 523, Dodoma, Tanzania
Email: ms_mathias@yahoo.co.uk

Allen R. Mushi
Department of Mathematics, University of Dar es salaam, Box 35062, DSM, Tanzania
Email: allenmushi66@gmail.com

Abstract—Given a weighted complete digraph, the Linear Ordering Problem (LOP) consists of finding and acyclic tournament with maximum weight. It is sometimes referred to as triangulation problem or permutation problem depending on the context of its application. This study introduces an algorithm for LOP and applied for triangulation of Tanzanian Input-Output tables. The algorithm development process uses Great Deluge heuristic method. It is implemented using C++ programming language and tested on a personal computer with 2.40GHZ speed processor. The algorithm has been able to triangulate the Tanzanian input-output tables of size 79×79 within a reasonable time (1.17 seconds). It has been able to order the corresponding economic sectors in the linear order, with upper triangle weight increased from 585,481 to 839,842 giving the degree of linearity of 94.3%.

Index Terms — Optimization, Linear Ordering, Input-Output Tables, Great Deluge Algorithm

I. Introduction

Linear Ordering Problem (LOP) involves finding an acyclic tournament in a complete weighted digraph with maximum weight. This is equivalent to finding a permutation of rows and columns (linear order) of the associated matrix such that the sum of the weights of the upper triangle is maximized, Mushi [1]. It is one of the combinatorial optimization problems that are classified as NP-hard (Non-deterministic Polynomial time) problems as discussed in Martí and Reinelt [2]. It is so classified because there is no general deterministic polynomial time algorithm for its solution that is known to date, Schiavinotto and Stützle [3].

Formally, given a complete digraph $D_n = (V_n, A_n)$ with n-nodes and arc weights x_{ij}, for all $i, j \in V_n$ (nodes set) and $(i, j) \in A_n$ (arcs set) defined on a positive weight function $x: A \mapsto \Re^+$, find an acyclic tournament $T \subseteq A_n$ such that the weight $x(T) = \sum_{(i,j)\in T} x_{ij}$ is the maximum possible [2].

The ideas on the solution of LOP existed since 1958, from the work done by Chenery and Watanable [4]. Since then, it has received considerable attention by many researchers and hence becoming a problem of interest to date. LOP is found in a number of applications, such as Triangulation of Input-Output matrices in economics, Archaeological seriation, Minimizing total weighted completion time in one-machine scheduling, Aggregation of Individual preferences, Grötschel, Junger and Reinelt [5], ordering of teams in sports tournaments, Mushi [1] as well as Machine translation, Tromble [6].

A number of algorithms for the LOP solution have been developed, however each algorithm works with some weaknesses depending on the level of complexity of the problem. Grötschel, Junger and Reinelt [5], introduced an exact algorithm which combines heuristics, cutting plane and branch and bound techniques basing on investigations of the LOP polytope. The algorithm was able to solve up to 60×60 Input-Output tables from the Linear Ordering Library (LOLIB) instances [7], with 83.185% degree of linearity and running time of 13 minutes and 25 seconds. Likewise, an exact algorithm focused on cutting plane and branch and bound procedures has been used to LOP in Mushi [1]. The approach relaxed integer constraints and solved the problem as a continuous Linear Programming problem with normal simplex algorithm and then applying cutting planes with available facets followed by branch and bound technique to get integral solution. The algorithm was able to solve the 41×41 Irish input-output table to optimality within reasonable time.

As pointed earlier, this is an NP-Hard problem and therefore no optimal algorithm is known that can solve a general problem within reasonable time. Consequently, a lot of effort has been devoted into finding good heuristic solutions. Although heuristics cannot give a guarantee of optimal solutions, they have been widely used to give good solutions within reasonable time. Marti and Reinelt [2] reports on the development of a heuristic, commonly referred to as Beckers method, based on calculations of quotients which were used as basis for ranking the economic sectors of an input-output table. However, the authors admit that their algorithm does not necessarily lead to good approximate solution to their triangulation problem.

Chanas and Koblański [8] designed a heuristic algorithm which could determine the solution for both maximization and minimization of the given problem criterion function, basing on the resulting permutation by sorting through insertion and permutation reversals. Experimental results show that the algorithm was able to give near optimal solutions in most instances. However, the method requires the problem to be decomposable into components, where methods for effective decomposition are difficult to find.

Garcia et al [9] developed a Variable Neighbourhood Search (VNS) algorithm. The algorithm combines various neighbourhoods for effective exploration of search space. The VNS results competed with well known heuristics such as Tabu Search but require further extensive studies. Campos, et al [10] presented meta-heuristic approaches for LOP, using scatter search technique. The approach used combined solution vectors that had proved effective in a number of problem settings, but was able to match only 30 out of 49 solutions of world problem instances in the LOLIB.

Recent work includes a publication by Celso and Mutsunori [11] on local algorithms. They provided improvements into the local search by introducing two algorithms namely LIST and TREE for neighbourhood structure. Computational experiments with random problems showed good results for sparse instances with density up to 10%. Even more recent is the 2014 publication by Tao et al [12], on Multi-Parent Memetic algorithm, denoted by MPM. The algorithm integrates particular features such as multi-parent recombination operator, a diversity based parent selection strategy and a quality-and-diversity based pool updating strategy. Computation results shows that the MPM is an efficient algorithm and outperforms previous Memetic algorithm in detecting lower bounds for challenging instances.

This paper introduces another heuristic algorithm for the LOP particularly for a triangulation problem using Great Deluge Algorithm (GDA) described in Doeck [13]. We are interested in the use of GDA particularly for Tanzanian Input-Output economic tables since the technique has been effective in other combinatorial optimization problems, including timetabling, Mushi [14], [15], Landa-Silva and Obit [16], and Turabie, Abdullah and McCollum [17]. Other successful applications include Dynamic Scheduling on Grid environment problem reported in McMullan and McCollum, [18].

The paper is organized as follows; first we give a mathematical formulation of the problem; secondly we present the Great Deluge Algorithm with the specific adaptation to the Linear Ordering Problem, including development of initial solution, selection of moves and neighbourhood structure. We then provide our analysis of results as applied to the Tanzanian Input-Output tables; and lastly we present conclusion and suggest areas for further research.

II. Mathematical Formulation

The LOP is formulated as a binary Integer Programming problem. The input-output table decision variable is defined as follows:-

Let

$$x_{ij} = \begin{cases} 1 & \text{if there is an arc } (i, j) \text{ between nodes } i \text{ and } j \\ 0 & \text{Otherwise} \end{cases}$$

$$\forall (i, j) \in A_n$$

Thus, the Linear Programming model as defined by Marti and Reinelt [2] is:

$$Maximize \ \sum_{(i,j)\in A_n} w_{ij} x_{ij} \tag{1}$$

Subject to:

$$x_{ij} + x_{ji} = 1, \forall i, j \in V_n$$

$$x_{ij} + x_{jk} + x_{ki} \le 2, \forall i, j, k \in V_n, j \ne k$$

$$x_{ij} \in \{0,1\}, \forall i, j \in V_n$$

Where w_{ij} represents the actual weights from the input-output table.

In Graphical representation, we define a digraph $D_n := (V_n, A_n)$ and find an acyclic tournament $T \subseteq A_n$ with the linear ordering $< V_{i1}, V_{i2}, ..., V_{in} >$ that gives the maximum sum of weights assigned to its corresponding arcs [2].

The most common exact approach in solving LOP is the use of a branch and cut algorithm that combines both cutting planes and branch and bound methods. This is facilitated by the development of deepest possible cutting planes known as facets that can prune infeasibilities from the relaxation of the associated linear programming problem. The known facets include the 3-dicycles, 3-fences, Möbius ladders of type M and type $\bar{M}$ [2]. These facets together with the minimal equation system give the following Linear Programming relaxation:

$$Maximize \ \sum_{\substack{i,j \\ i \ne j}} c_{ij} x_{ij} \tag{2}$$

Subject to:

$$x_{ij} + x_{ji} = 1, \text{ for all } 1 \le j \le j \le n,$$

$$x_{ij} \ge 0, \text{ for all } 1 \le i, j \le n, i \ne j,$$

$$x(C) \le 2, \text{ for all 3-dicycles } C \text{ in } A_n,$$

$$x(F) \le 7, \text{ for all 3-Fences } D = (V, F) \text{ in } D_n,$$

$$x(M) \le 8, \text{ for all Möbius ladders } D = (V, M) \text{ of type } M_2 \text{ or } \bar{M}_2 \text{ in } D_n$$

It has been shown in [2] that this problem has $\binom{n}{2}$ equations, $n(n-1)$ non-negativity constraints, $2\binom{n}{3}$ 3-

dicycle inequalities, $120\binom{n}{6}$ 3-fence inequalities, and $360\binom{n}{6}$ Möbius ladder inequalities.

Due to this enormous number of constraints, it is impractical to list all the constraints and solve the linear program using available computer code. Instead, a cutting plane algorithm is applied where facets are added to the problem one at a time until the solution is found or the problem size has been reduced to an extent that it can be solved by branch and bound method.

However, this approach has been only successful in solving specific instances to optimality due to large size of the problem. The use of heuristics is therefore a preferred approach when it comes to large problem sizes and hence the choice of Great Deluge heuristic approach in this work.

A. Linear Ordering as a Triangulation Problem

As described by Marti et al in [7], given an (n, n) matrix $C = (c_{ij})$, the triangulation problem involves the determination of a simultaneous permutation of the rows and columns of the matrix C such that the sum of upper-diagonal entries is as large as possible.

By setting arc weights $w_{ij} = c_{ij}$ for the complete digraph D_n, the triangulation problem for C can be solved as Linear Ordering Problem for D_n. Conversely, a linear ordering problem for D_n can be transformed into a triangulation problem for an (n, n) matrix $C = (c_{ij})$ by setting $c_{ij} = w_{ij}$ and the diagonal entries $c_{ii} = 0$.

Example

Consider the following directed compete graph with weights on arcs (Fig. 1);

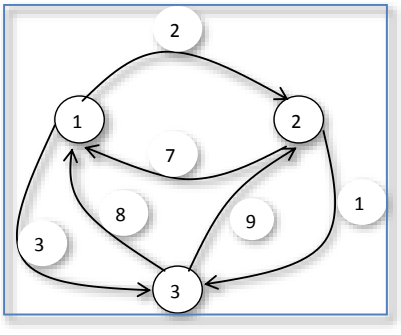

Fig. 1. Complete digraph example

The associated matrix is as shown in Table 1;

Table 1. Input-output table example

	1	2	3
1	0	2	3
2	7	0	1
3	8	9	0

Sum of upper weight = 2+3 +1 = 6

An acyclic tournament with the highest weight in this simple example is achieved by picking arcs with highest weight which does not violate the acyclic condition and covers all arcs. The graph below gives the best solution (Fig. 2);

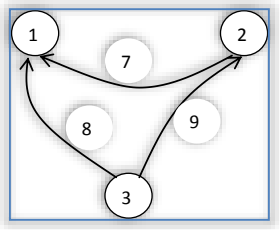

Fig. 2. Acyclic tournament with highest weight example

Ordering procedure

The first node is the one with no entering arc (i.e. 3); the last node is the one with no leaving arc (i.e. 1). The order is therefore: 3, 2, 1 and the associated matrix is as shown in Table 2;

Table 2. Triangulated Input-Output table example

	3	2	1
3	0	9	8
2	1	0	7
1	3	2	0

Sum of upper weight = 9 + 8 + 7 = 24

Our case study problem is the input-output table for Tanzanian Economy with 79 sectors of economy and cannot be solved by such a simple procedure. Triangulation is an important factor in understanding complex series of interactions among sectors of any county's economy [19].

III. GDA Algorithm Formulation

One of the main challenges associated with global heuristic techniques such as Genetic Algorithms, Tabu Search, Simulated Annealing and many others is the sheer number of parameters that have to be selected and their sensitivity towards the choice of the best solution. Great Deluge algorithm was designed to address this problem of multiple parameters by minimizing the number of parameter requirements without jeopardising the quality of solution. The algorithm was introduced by Dueck, G. [13] and in general it requires only one parameter.

The idea is a simulation of an object in a mountainous space which is under pouring rain. The object wonders randomly on the space, but there is water level below which it can't go because of water. If this level is L, then the object accepts any area that has a value greater than L. As time goes on, L rises slowly and finally forced up onto a peak (and then the rain stops). The idea can easily be defined for the minimization case as shown in the pseudo code in Fig 错误!未找到引用源。 3. This simulation is compared with the Noah's Ark and hence the name Great Deluge.


```
Great Deluge Algorithm
Set the Initial Solution So;
Calculate Initial cost f(So);
Initial Level L = f(So);
Specify input parameter ΔL;
While no further improvement possible {
        Define neighbourhood N(So)
Randomly select a candidate solution
S∈N(So);
        If f(s) ≤ f(So)
                Accept new solution (So = S);
        Else if f(s) ≤ L)
                Accept new solution (So=S);
        Else
                Reject new solution
                }
        Lower Level L = L - ΔL;
        }
So is the best solution;
End Great Deluge
```

Fig. 3. Great Deluge Algorithm's Pseudo-code

In this minimization case the algorithm allows a reduction in solution values according to their improvement. However, the approach also accept worse candidate solution if its value is less than or equal to the given upper limit *L*. The main function of L during search process is to restrict some of the search space and thereby forces the current solution to escape into the remaining feasible space. It can be noted that the user only needs to decide one input parameter i.e. ΔL which controls the reduction of level *L*.

Initially, the controlling process is slow, and in this case the value of *L* does not exceed the current solution, but only prohibits the longest backward moves [20]. The situation progresses until the value of *L* exceeds the current solution, and in this lower level, only good moves are accepted.

A. Initial Solution

Initially, we need a quick feasible solution as an input to the GDA. The initial solution in this case is easily found by picking all upper triangle values of the solution matrix. In this case we are guaranteed that the solution does not contain any dicycles and covers all nodes of the associated graph. Therefore the initial solution is set as $S_0 = \left(x_{ij}^0\right)$ Such that:

$$x_{ij}^0 = \begin{cases} 1 & for \quad all \quad i < j \\ 0 & otherwise \end{cases}$$

Objective function value for the initial solution is the sum of the product of weights and the associated binary values in the solution matrix and is represented by $f\left(S_0\right) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} w_{ij}x_{ij}^0$

Where w_{ij} is the weight of arc (i, *j)* in the input-output table.

B. Moves and Neighbourhood structure

The algorithm uses swap moves which involve swapping of arcs in the graph. Two nodes *i* and *j* are selected at random in the current solution and swapped into *j* and *i* in the order. The corresponding binary values are therefore swapped as well, since only one of the (*i*,*j*) and (*j*,*i*) can have a value of 1 at the same time in the solution. After this choice the algorithm check if there is violation of constraints. If no violation, the swapping is confirmed otherwise no swapping is done. The process continues until a swap with no violation of constraints is found.

C. Checking violation of constraints

Given the choice of the initial solution, it guarantees that only one of $x_{ij} = 1$ at the same time, and therefore the constraint $x_{ij} + x_{ji} = 1, \forall i, j \in V$ is satisfied. Furthermore, the selection of a swap move guarantees that the constraint will always be satisfied. The only constraint to be checked for violation is the dicycle constraint. Given two nodes say *i* and *j* which have been picked randomly for swapping, and index $k = \{1, 2, ..., n-2\}$ is defined and the following is checked for any violation: $x_{ij} + x_{jk} + x_{ki} \le 2, \forall i, j, k \in V_n, j \ne k$. The algorithm stops once the first violation is detected and returns a violation indicator; otherwise the swapping process is confirmed.

D. Increasing level rate (ΔL);

As shown in Fig. 3, Great Deluge algorithm is designed to accept good solutions but can accept bad moves only when the function value is greater than a specified level value (Level). Given an initial solution $f\left(S_0\right)$, the increasing level rate is calculated as follows;

$\Delta L = \dfrac{f(S_0)}{N_{mov}}$, where $\left(Nmov\right)$ is the pre-determined number of moves and the only input parameter. Initially *Level* is assigned the values of the initial solution, it is then steadily increased by ΔL at each iteration.

IV. Analysis of Results

The algorithm code was implemented on a C++ programming language and tested on the personal computer with 2.40GHZ speed processor. It has been tested on the Tanzanian input-output table. Tanzanian input-output economic table is classified into 79 main sectors of economy based on the type of product produced in each unit according to Tanzanian Central Bank (CB) data of 1992 [21]. This therefore gives s 79×79 size of a matrix. The algorithm was tested by different values of $\left(Nmov\right)$ and the solution obtained after a number of iterations (MaxIters) as shown in Table 3. The number of MaxIters was varied from 0 to 44,000 with fixed interval of 2,000. The results were recorded

three times at different number of moves $(Nmov)$ i.e. 10,000, 30,000, and 60,000.

Table 3. Algorithm Performance Results

MaxIters	S1(10,000)	S2(30,000)	S3(60,000)
0	585,481	585,481	585,481
2,000	677,018	623,765	608,260
4,000	753,655	682,803	641,191
6,000	802,245	702,740	649,524
8,000	816,925	741,428	664,371
10,000	830,979	778,743	683,898
12,000	838,002	813,608	704,031
14,000	839,256	828,143	727,310
16,000	839,787	831,331	746,432
18,000	839,787	833,948	762,396
20,000	839,827	836,710	779,867
22,000	839,827	838,279	799,089
24,000	839,842	839,657	816,862
26,000	839,842	839,720	825,097
28,000	839,842	839,730	833,667
30,000	839,842	839,842	835,878
32,000	839,842	839,842	838,126
34,000	839,842	839,842	838,168
36,000	839,842	839,842	838,886
38,000	839,842	839,842	839,695
40,000	839,842	839,842	839,838
42,000	839,842	839,842	839,842
44,000	839,842	839,842	839,842

The speed of conversion does not necessarily increase with the number of moves. As can be seen in Table 3, the algorithm converged faster with the lower number of moves (10,000), signifying that there is a threshold of number of moves that can save time in finding a good solution. This is clearly demonstrated in Fig. 4, where both values of Nmov converged to the same solution but Nmov=10,000 converged faster i.e. at a lower number of iterations.

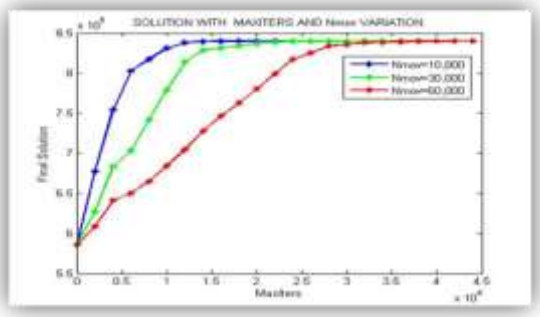

Fig. 4. GDA Performance

It was also interesting to note the behaviour of the solution in relation to changes in the value of levels in the Great Deluge procedure. A specific case was picked with the parameters shown in Table 4 and observes all iterations until convergence to a particular solution.

Table 4. Parameters under a single run

Nmov	MaxIters	Time (Seconds)
60,000	42,000	1.172

The results are as shown in Fig. 5, where the solution values are fluctuating above the level line. Initially there is high fluctuation showing the high acceptance of bad moves but later the solution stabilizes and finally converges regardless to the increase the level values. This clearly demonstrates the expected performance of Great Deluge and the influence of the level parameter in the quality of solution.

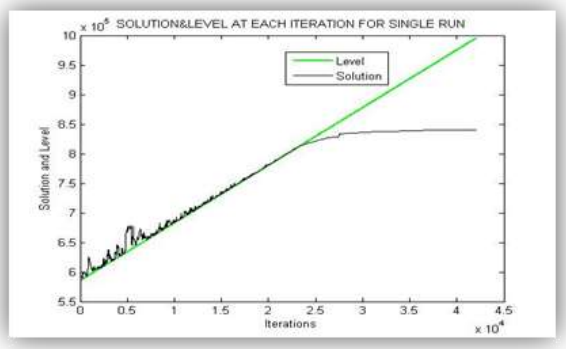

Fig. 5. Solution-Level relationship

A. Degree of Linearity and Order of sectors

Degree of linearity is an index that shows the extent of triangulation of the matrix. This is given by

$$\lambda = \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{ij}}{\sum_{i=1}^{n}\sum_{j\neq i}^{n} w_{ij}} \qquad (3)$$

This is basically the ratio of the sum of the weights above the diagonal to the sum of all weights in the matrix (except diagonals). The value of λ=1 for a perfectly linear economy, Leontief [22]. The computation of the degree of linearity (λ) for our solution is shown in Table 5.

Table 5. Measure of Linearity

Sum of I/O table entries for $i < j$	Sum of I/O table entries for $i \neq j$	Degree of Linearity
839,842	890,629	94.3%

The degree of linearity attained so far (94.3%) shows how well the input-output table for this particular problem is triangulated. The original sectors of the economy in the input-output tables were ordered as shown in the table 6.

After running the Great Deluge Algorithm the results are shown in Table 7 where the order has been completely changed to reflect a maximization of the upper diagonal entries of the input-output table.

Table 6. Prior ordering

1. Growing of maize	41. Manufacture of tobacco products
2. Growing of paddy	42. Spinning and finishing of textiles
3. Growing of sorghum/millets	43. Manufacture of made-up textiles
4. Growing of wheat	44. Manufacture of cordage, rope and twine
5. Growing of beans	45. Manufacture of wearing apparel
6. Growing of cassava	46. Manufacture of other textiles
7. Growing of other cereals	47. Manufacture of leather products
8. Growing of oil seeds	48. Manufacture of footwear
9. Growing of other roots and tubes	49. Manufacture of wood and wood products
10. Growing of cotton	50. Manufacture of pulp and paper
11. Growing of coffee	51. Printing and publishing
12. Growing of tobacco	52. Manufacture of basic and industrial chemicals
13. Growing of tea	53. Manufacture of fertilizers and pesticid.
14. Growing of cashewnuts	54. Petroleum refineries
15. Growing of sisal fibre	55. Manufacture of rubber products
16. Growing of coconuts	56. Manufacture of plastic products
17. Growing of sugar cane	57. Manufacture of glass and glass products
18. Growing of bananas	58. Manufacture of cement and clay
19. Growing of other fruits	59. Processing of iron, steel and non-ferrous metals
20. Growing of other vegetables	60. Manufacture of metal products
21. Growing of other oil seeds	61. Manufacture of machinery and equipment
22. Growing of other crops	62. Manufacture of electrical equipment
23. Operation of poultry	63. Manufacture of transport equipment
24. Fishing and fish farms	64. Manufacture of other goods
25. Other farming of animals	65. Production and distribution of electricity
26. Hunting and game propagation	66. Collection and distribution of water
27. Forestry and logging	67. Construction
28. Quarrying of stone, clay and sand	68. Wholesale and retail trade
29. Extraction of salt	69. Hotels and restaurants
30. Mining of gemstones	70. Land transport
31. Other mining and quarrying	71. Other transport
32. Processing of meat and dairy products	72. Post and telecommunication
33. Canning and pressing of fruits and vegetables	73. Financial intermediation
34. Manufacture of oils and fats	74. Real estate
35. Grain milling	75. Business service activities
36. Manufacture of bakery products	76. Public administration
37. Manufacture of sugar and confectionery	77. Education service activities
38. Manufacture of other food products	78. Health service activities
39. Local brewing activities	79. Other service activities
40. Manufacture of beverages	

Table 7. Post ordering

65. Production and distribution of electricity	10. Growing of cotton
72. Post and telecommunication	20. Growing of other vegetables
75. Business service activities	21. Growing of other oil seeds
68. Wholesale and retail trade	22. Growing of other crops
66. Collection and distribution of water	37. Manufacture of sugar and confectionery
73. Financial Inter-mediation	51. Printing and publishing
70. Land transport	15. Growing of sisal fibre
49. Manufacture of wood and wood products	28. Quarrying of stone, clay and sand
54. Petroleum refineries	33. Canning and pressing of fruits and vegetables
52. Manufacture of basic and industrial chemicals	43. Manufacture of made-up textiles
50. Manufacture of pulp and paper	7. Growing of other cereals
67. Construction	17. Growing of sugarcane
79. Other service activities	23. Operation of poultry
56. Manufacture of plastic products	45. Manufacture of wearing apparel
53. Manufacture of fertilizers and pesticid	18. Growing of bananas
64. Manufacture of other goods	30. Mining of gemstones
74. Real estate	31. Other mining and quarrying
60. Manufacture of metal products	39. Local brewing activities
44. Manufacture of cordage, rope and twine	58. Manufacture of cement and clay
59. Processing of iron, steel and non-ferrous metals	71. Other transport
61. Manufacture of machinery and equipment	8. Growing of oil seeds
14. Growing of cashew nuts	13. Growing of tea
63. Manufacture of transport equipment	34. Manufacture of oil and fats
5. Growing of Beans	19. Growing of other fruits
25. Other farming of animals	29. Extraction of salt
55. Manufacture of rubber products	35. Grain milling
9. Growing of other roots and tubes	24. Fishing and fish farms
11. Growing of coffee	32. Processing of meat and dairy products
16. Growing of coconuts	38.Manufacture of other food products
3. Growing of sorghum/ Millets	36. Manufacturing of bakery products
26. Hunting and game propagation	6. Growing of cassava
27. Forestry and logging	46. Manufacture of other textiles
42. Spinning and finishing of textiles	78. Health service activities
47. Manufacture of leather products	12. Growing of tobacco
57. Manufacture of glass and glass products	40. Manufacture of beverages
1. Growing of maize	77. Education service activities
48. Manufacture of foot wear	41. Manufacture of tobacco products
62. Manufacture of electrical equipment	69. Hotels and restaurants
2. Growing of paddy	76. Public administration
4. Growing of wheat	

V. Conclusion and Further Research

This paper presents a Great Deluge Algorithm as an approach in solving the Linear Ordering Problem. This is a case study of Tanzanian input-output table. We have been able to implement the algorithm and obtain a linear order with degree of linearity of 94.3%. This demonstrates that Great Deluge is a good heuristic for the Linear Ordering problem.

Since the algorithm was tested on a Tanzanian case study, it is worth testing the performance of the algorithm on benchmark problems in the LOLIB library which have been tested with other algorithms and compare results. However fast the algorithm is, it is still not guaranteed that the obtained solution is optimal. A lot of effort has been devoted to development of exact methods especially in the identification of unique facets for the problem. It is therefore worth investigating further the use of exact methods for the Tanzania Input-output table.

References

[1] Mushi Allen Rangia, "The Linear Ordering Problem; An Algorithm for the Optimal Solution", *African Journal of Science and Technology*, Vol. 6, pp. 51-64, 2005.

[2] Martí R and Reinelt G, "The Linear Ordering Problem, Exact and Heuristic Methods in Combinatorial Optimization", Springer, Applied Mathematical Sciences, 2011.

[3] Schiavinotto T and Stützle T, "The Linear Ordering Problem, Instances, Search Space Analysis and Algorithms", *Journal of Mathematical Modelling and Algorithms*, Vol. 3, pp. 367-402, 2004.

[4] Chenery BH and Watanable T, "International Comparisons of the Structure of Production", *Econometrica*, Vol. 26, pp. 4, 1958.

[5] Grötschel M, Jünger M and Reinelt G, "A Cutting Plane Algorithm for the Linear Ordering Problem", *Operation Research Society of America*, Vol. 32, pp. 3206-1195, 1984.

[6] Tromble WR, "Search and Learning for the Linear Ordering Problem with an Application to Machine translation", Johns Hopkins University, USA (PhD thesis), 2009.

[7] Martí, Reinelt and Duarte, "Linear Ordering Problem", Optsicom Project: http://www.optsicom.es/lolib/, June 2014.

[8] Chana S, Kobylański P, 1996, "A New Heuristic Algorithm Solving the Linear Ordering Problem". *Kluwer Academic Publishers*, Vol. 6, pp. 191-205, 1996.

[9] Garcia C, Pérez-Brito D, Campos V and Martí R, 2005, "Variable Neighborhood Search for the linear ordering problem", *Elsevier Science Ltd*, Vol.33, pp. 3549–3565, 2006.

[10] Campos V, Glover F, Laguna M and Martí R, "An Experimental Evaluation of Scatter Search for Linear Ordering Problem", *Journal of Global Optimization*, Vol. 21, pp. 397-414, 2001.

[11] Celso S. Sakuraba, Mutsunori Yagiura, "Efficient Local Search Algorithms for the Linear Ordering Problem", International Transactions in Operational Research, Vol. 17, pp 711-737, 2010.

[12] Tao Y., Wang T, Lu Z., Hao J., "A Multi-Parent Memetic Algorithm for the Linear Ordering Problem", arxiv.org/abs/1405.4507v1 [cs.NE] (Submitted to arXiv on 18, May 2014).

[13] Doeck G, "New Optimization Heuristics, The Great Deluge Algorithm and The Record-to-Record Travel", *Heidelberg Scientific Centre*, Vol. 104, pp. 86-92, 1993.

[14] Mushi Allen Rangia, "Two-Phase Great Deluge Algorithm for Course Timetabling problem", *International Journal of Advanced Research in Computer Science*, Vol. 2, No.5, pp. 73-83, 2011.

[15] Mushi Allen Rangia, "Non-Linear Great Deluge algorithm for Tanzanian High Schools Timetabling", *International*

Journal of Advanced Research in Computer Science, Vol. 2, No. 4, pp.584-590, 2011.
[16] Landa-Silva D and Obit J, "Great Deluge with Non-linear Decay Rate for Solving Course Timetabling Problems", *Intelligent Systems*, Vol. 1, pp. 8–18, 2008.
[17] Turabieh H, Abdullah S and McCollum B, "Electromagnetism-like Mechanism with Force Decay Rate Great Deluge for the Course Timetabling Problem", Springer, UK, 2009.
[18] McMullan P and McCollum B, "Dynamic Job Scheduling on Grid Environment Using Great Deluge Algorithm", Berlin Heidelberg, Springer, Vol. 4671, pp. 283–292, 2007.
[19] Lamel J., Richter J., Teufelsbauer W., "Patterns of industrial structure and economic development: Triangulation of input-output tables of ECE countries", European Economic Review, Vol. 3, No. 1, pp 47–63, 1972.
[20] Burke E., Yuri b., Newall J., Petrovic, S., "A Time Predefined Local Search Approach to Exam Timetabling Problems", IIE Transactions of Operations Engineering, Vol. 36 pp 509-528, 2004.
[21] National Bureau of Statistics, "Input-Output Table of Tanzania for 1992", Input-output table construction project, supported by Sida through Statistics Sweden Vol. 2, 1992.
[22] Leontief, Wassily, "Input-Output Economics", Oxford University Press, New York, USA, 1986.

Authors' Profiles

Amos Mathias: A postgraduate student in the Masters of Science in Mathematical modeling programme at the University of Dar es Salaam in Tanzania. Mathias is also working in the Department of Science Mathematics and Technology Education at the University of Dodoma in Tanzania.

Allen Rangia Mushi: An Associate Professor of Mathematics from the Department of Mathematics of the University of Dar es Salaam in Tanzania. His area of interest is Operations Research specifically the Combinatorial Optimization Problems. Major research work includes Timetabling, Linear Ordering, Resource Leveling in manufacturing industries, Municipal Solid Waste Disposal models and Graph Theory.

How to cite this paper: Amos Mathias, Allen R. Mushi,"Great Deluge Algorithm for the Linear Ordering Problem: The Case of Tanzanian Input-Output Table", International Journal of Information Technology and Computer Science(IJITCS), vol.7, no.7, pp.28-34, 2015. DOI: 10.5815/ijitcs.2015.07.04