Programmatic Implementation of the Karmarkar's Algorithm for Vacation Package Synthesis Process Optimization

H.Ganepola[1], J. Padukka[2], R. Prasad[3], L. Samarakoon[4]
Department of Computer Science and Engineering, Faculty of Engineering, University of Moratuwa, Sri Lanka
hasini.ganepola@uom.lk[1], janaka.padukka@uom.lk[2], rajeev.prasad@uom.lk[3], lahiru.samarakoon@uom.lk[4]

Abstract

Vacation Packages are a basic component of the modern travel industry. Putting together a set of vacation packages, which can generate the highest revenue to the tour operator, is a hectic task which will need a lot of man hours.
In this paper, we present a linear programming approach which ensures the maximum revenue to the tour operator, where all the contract constraints are satisfied. From LP model formulation to the solving algorithm and its results will be presented in this paper. Karmarkar's Projective Scaling algorithm, which is the solving algorithm will be explained in the process, with its implementation approach

1. Introduction

Mathematical Optimization techniques are highly utilized in modern revenue based global economy, to ensure the healthy running of the businesses. In this paper, our focus is to introduce such a scenario in travel industry, where the mathematical optimization technique, widely known as Linear Programming, is utilized to ensure high profits for the tour operators.

Majority of the linear programming problems are solved mainly using the primal/dual simplex method due to the underlying linear programming structure. And also with the increase of the no. of variables in the problem, simplex methods complexity becomes exponential. First polynomial time algorithm to solve a LP problem was introduced in 1979 by L.G. Khachiyan. But with Khachian's method[4],[5] is computational experience with it and its variants has been very disappointing. This has contributed to the realization that one ought to consider as "efficient" only low-order polynomial algorithms, perhaps those which are also strongly or genuinely polynomial. The practical performance of the Khachian's algorithm is strongly connected with its theoretical worst-case bound. Magnitude of the input data is connected with the efficiency of it. It has been found that even problems with about 100 variables can require an enormous amount of effort. In 1984, a new algorithm was presented by N. Karmarkar, which is an interior point method[1] like Khachiyan's one with a polynomial time complexity $O(n^{3.5}L)$[4]defeating $O(n^{6}L)$.[5]

Khachian's ellipsoid method and Karmarkar's projective scaling method seek the optimum solution to an LP problem by moving through the interior of the feasible region. A schematic diagram illustrating the algorithmic differences between the Simplex and the Karmarkar's algorithm is shown in figure 1. Khachian's ellipsoid method approximates the optimum solution of an LP problem by creating a sequence of ellipsoids (an ellipsoid is the multidimensional analog of an ellipse) that approach the optimal solution. [3]

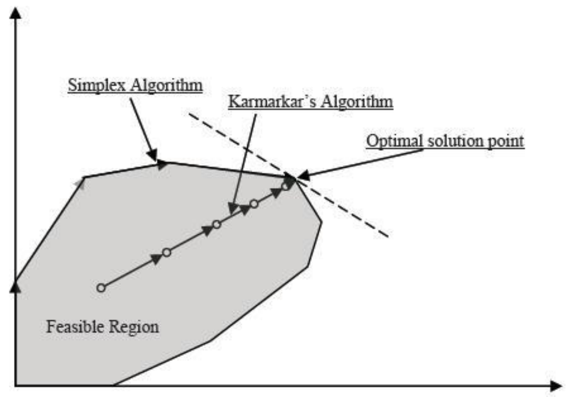

Figure 1- Difference in optimum search path between Simplex and Karmarkar's Algorithm

Both Khachian's ellipsoid method and Karmarkar's projective scaling method have been shown to be polynomial time algorithms. This means that the time required to solve an LP problem of size *n* by these two methods would take at most where an^{b} and *b* are two positive numbers.

On the other hand, the Simplex algorithm is an exponential time algorithm in solving LP problems. This implies that, in solving an LP problem of size *n* by Simplex algorithm, there exists a positive number c such that for any n, the Simplex algorithm would find its solution in a time of at most $c2^n$. For a large enoug n (with positive a, b and c), $c2^n > an^b$. This means that in theory, the polynomial time algorithms are computationally superior to exponential algorithms fo large LP problems.[3]

In this paper our focus it to model and implement th solver for the linear programming problem, which wil be explain in the next section, by the computationally much feasible Karmarkar's Algorithm.

2. Background

Travel industry is a very expanded and highl revenue generating economic entity in most of th countries around the world. Supply chain of the trave industry consists of the suppliers such as airlines hotels and transfer companies which provide their inventory to the tour operators, where they sold them to the tour agencies as vacation packages to be sold to the customers. The vacation packages are a combination of elements and the permutations that can be achieved using the different options in the contracts that are signed between suppliers and tour operators.

In order to put up a set of vacation packages, which would earn the highest revenue to the tour operator, is a very important step. For that purpose, tour operators have setup product planning departments, which needs a great deal of human intuition as well as the historical data on the passenger interests.

The objective of this paper is to find the optimum combination of elements from the various available contractual elements such that the maximum yield and revenue can be achieved using the linear programming approach for the optimization.

2. LP Model of Vacation Package Synthesis Process

In this section, the main idea is to formulate the linear programming problem, which incorporates with the vacation package synthesis process. For simplicity, the model will be explained using limited no. of vacation package components.

Let's consider a scenario where, one departure city, two airlines and two destination cities with two hotels in each city. Figure 2, shows the no. of vacation package combinations, which can be built using these components.

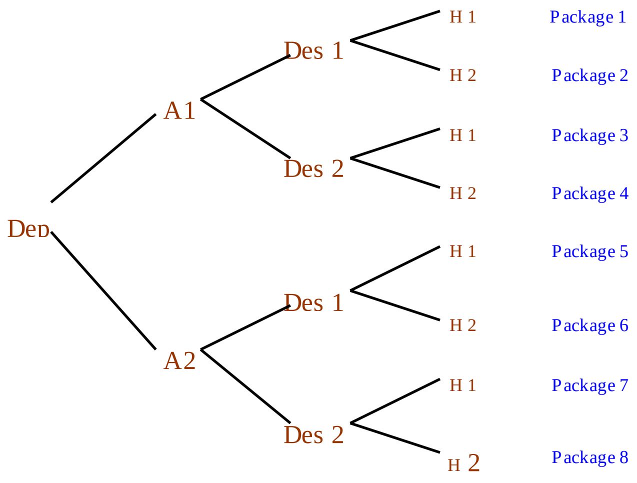

Figure 2- Vacation Package Generation

Considering contract information incorporated with the components and vacation package combinations. The linear programming problem can be formulated in two steps.

2.1 Formulation of Constraints

Tour operators are contracted with suppliers on the vacation package components. Therefore the no. of vacation packages generated using specific components will be constrained.

For example,

Let $S_{D,A,Dest,d}$ be the no.of seats available on flight A, which departs from D, to destination Dest on day d.
And
Let $X_{D,A,Dest,H,d}$ be the no. of passenger who are willing to travel on airline A, from departure D to destination Dest on day d and want to stay at hotel H.

Let's assume the packages are single. And there for the no of packages will be equal to the no. of passengers.

Now let's consider the airline A1, from departure city Dep to Dest1. As the no. of occupants for airline A1, for day d from Dep to Dest1 should be less than or equal to the no. on the contract, using the same notation we can say,

$$X_{Dep,A1,H1,d} + X_{Dep,A1,H2,d} <= S_{Dep,A1,Dest1,d}$$

Above constraint represents the supplier contract for airline A1. In this manner, constraints can be formulated for the other package components as well. The unique combination of Dep,A1,H1,d will represents the incorporated package in this scenario.

2.2 Formulation of the Objective Function

The main idea of the objective function is to maximize the overall profit gained by the generated vacation packages.

Let's consider the vacation package Dep,A1,Dest1,H1 used in constraint formulation and denote it as P1.

Then,
The cost of P_1 = Hotel Cost + Flight Cost

The profit of P_1 = Profit Margin* The cost of P1

Let p_1 be the marginal probability of buying package P1 and x_1 be the no. vacation package allocations.

Then for packages P1 to P8, the total profit can be obtained by,

$$Total\ Profit = \sum_{i=1}^{8} p_i * Profit\ of\ P_i * x_i$$

The objective function for optimization will be the above one. That is Maximize Total Profit, subjected to constraints of the form explained in the previous section.

3. Karmarkar' Algorithm

This section focuses on the details of the Karmarkar's Algorithm, which we used as the solving algorithm and the transformation algorithm of the standard LP problem formulated in the previous section to the Karmarkar's canonical form.

3.1 Algorithm and Its Canonical Form

Karmarkar's projective scaling method starts with a trial solution and shoots it towards the optimum solution.

To apply the Karmarkar's algorithm, LP problem should be expressed in the following form.

Minimize $\quad Z = C^T X$

Subjected to: $\quad AX = 0$
$\quad \mathbf{1}X = 1$

With: $\quad X \geq 0$

$$\text{Where,} X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, C = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}, \mathbf{1} = [1 \quad 1 \quad \ldots \quad 1]_{(1\times n)}$$

$$A = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} \ and\ n \geq 2$$

It is assumed that, $X_0 = \begin{bmatrix} 1/n \\ 1/n \\ \vdots \\ 1/n \end{bmatrix}$

is a feasible solution and $Z_{min} = 0$, and the two other variables are defined as,

$$r = \frac{1}{\sqrt{n(n-1)}}, \alpha = \frac{(n-1)}{3n}$$

Iterative steps are involved in Karmarkar's Algorithm to find the optimal solution ,

In general, kth iteration involves following computations.

1. Compute $C_p = [I - P^T(PP^T)P]\bar{C}^T$

   Where,

$$P = \begin{bmatrix} AD_k \\ 1 \end{bmatrix}, \bar{C} = C^T D_k \ and$$

$$D_k = \begin{bmatrix} X_k(1) & 0 & 0 & 0 \\ 0 & X_k(2) & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & X_k(n) \end{bmatrix}$$

If $C_p = 0$, any feasible solution becomes and optimal solution. Further iteration is not required. Otherwise followings should be computed.

2. $Y_{new} = X_0 - \alpha r \frac{C_p}{\|C_p\|}$

3. $X_{k+1} = \frac{D_k\ Y_{new}}{1D_k\ Y_{new}}$, for k= 0, $\frac{D_k\ Y_{new}}{1D_k\ Y_{new}} = Y_{new}$
   Thus, $X_1 = Y_{new}$

4. $Z = C^T X_{k+1}$

5. Repeat steps 1 to 4 by changing k=k+1[1],[2]

3.2 Transformation Logic

Let $Maximize\ \{cx: Ax = b, x \geq 0\}$ be the standard LP problem.

Here A is the LHS of the constraint matrix, with mxn dimension with rank m.

Then dual of the above can be given as,

$$Minimize\ \{bx: A^T w = b, w \geq 0\}$$

Objective equality constraint become,

$$cx - bw = 0, x \geq 0, w \geq 0$$

Introduction of bounding constraint to regularize the problem,

$$\sum x + \sum w \leq Q$$

Here Q may be taken as some known small integer bound on the sum of variables, derived from feasibility and/or optimality considerations. In the worst case, $Q = 2^L$ where L is the number of binary bits required to record all the data of the problem and is known as the input length of an instance of the problem.

$$L = [1 + \log(1+m)] + [1 + \log(1+n)] + \sum_j \{1 + [\log(1 + |c_j|)]\} + \sum_i \sum_j \{1 + 1 + [\log(1 + |a_{ij}|)]\} + \sum_j \{1 + [\log(1 + |b_j|)]\}$$

Adding this constraint along with a slack variable s_1, we may write the equation as follows.

$$\sum x + \sum w + s_1 = Q$$

Then the introduction of the dummy variable s_2 with the constraint, $s_2 = 1$ to rewrite the constraints, $Ax - bs_2 = 0$, $A^T w - cs_2 = 0, s_2 = 1$,

$$1x + s_1 + s_2 = Q + 1 \text{ and}$$

$$\sum x + \sum w + s_1 + Qs_2 = 0$$

Where, $x \geq 0, w \geq 0, s_1 \geq 0, s_2 \geq 0$

As the next step, we use a change of variables defined by,
$x_i = (Q+1)y_i\ and\ w_j = (Q+1)y_j$

This gives the system,

$$Ay - by_{2(m+n)+2} = 0,$$

$$A^T y - cy_{2(m+n)+2} = 0,$$

$$\sum_{i=1}^{n} c_i y_i - \sum_{j=1}^{m} \sum_{i=m+n+1}^{2m+n+1} b_j y_i = 0,$$

$$\sum_{i=1}^{2(m+n)+1} y_i - Qy_{2(m+n)+2} = 0,$$

$$\sum_{i=1}^{2(m+n)+2} y_i = 1, y_i \geq 0$$

Finally introducing the artificial variable $y_{2(m+n)+3}$ with coefficients such that the sum of the coefficients in each homogeneous constrain is zero and accommodating $y_{2(m+n)+3}$ in the final equality constrain, we get the problem in the Karmarkar's Canonical Form.

$$Minimize\ \ y_{2(m+n)+3}$$

Subjected to;

$$Ay - by_{2(m+n)+2} - [A1^T - b]y_{2(m+n)+3} = 0,$$

$$A^T y - cy_{2(m+n)+2} - [A^T 1 - b]y_{2(m+n)+3} = 0,$$

$$\sum_{i=1}^{n} c_i y_i - \sum_{j=1}^{m} \sum_{i=m+n+1}^{2m+n+1} b_j y_i - [\sum_{i=1}^{n} c_i - \sum_{j=1}^{m} b_j] y_{2(m+n)+3} = 0,$$

$$\sum_{i=1}^{2(m+n)+1} y_i - Qy_{2(m+n)+2} - [2(m+n) + 2 - Q]y_{2(m+n)+3} = 0,$$

$$\sum_{i=1}^{2(m+n)+3} y_i = 1$$

$$and\ y_i \geq 0, i = 1,2, \ldots ,2(m+n)+3\ [2]$$

4. Implementation Approach

Implementation of the solution can be divided into three major parts as,

- LP Problem Formulation
- Transformer Logic
- Solver Logic
- Retranformer Logic

In the LP Problem Formulation, the contract data extraction for the formulation of the vacation packages, constraints and the objective function are carried out. The outputs of this module are the three matrices which represent the standard LP problem, the objective matrix, constraint LHS matrix and the constraint RHS matrix.

On the reception of the above three matrices, transformer module's job is to convert it to the Karmarkar's Canonical Form, which can be solved by the Solver. The core logic of the transformer is described in section 3.2. Output of the transformer is the LP problem, which is in the Karmarkar's Canonical Form. In the implementation, the output is given as a matrix which can be handled easily by the Solver, which is highly capable of matrix operations.

Solver's logic is highly incorporated with matrix manipulations. Therefore in the implementation of the Solver, matrix operations are implemented in separate classes and the algorithm is implemented in the Solver class, which contains the procedural algorithm explained in section 3.1.

At last by the Retransformer, the solved solution matrix given by the Solver will be mapped into the original LP problem generated by the LP Problem Formulator. Output of this is used to reporting purposes.

5. Performance Improvement Techniques

In this section, idea is to present the techniques used in improving the performance of the overall system. These techniques include the algorithmic techniques as well as the implementation specific techniques related to java.

- **Matrix Representation and Operations**

Matrix representation of this system was done in object oriented manner. The matrix is represented within the class by using a double two dimensional array. Due to the memory structure of the java array, in traversing the matrix, it has be proven that row wise traversing takes less time than column wise traversing. Therefore in this implementation, in traversing the matrix, row wise approach is taken to ensure high performance.

The choice of the matrix inversion algorithm was another decision, where the performance can be degraded. In this implementation, in computing the matrix inverse LU and QR decomposition was used. LU decomposition was used when the matrix is square and QR was used when it is used. For a square matrix of nxn, LU decomposition is speeds up calculation by n/4 times than the famous Gaussian Elimination. [7]

- **Multi threading**

In the Transformation process, formulation of the Karmarkar's constrain matrix is implemented concurrently with the help of multi threading. Multi Threading the sequential algorithm of transforming the general linear programming problem into Karmarkar's canonical form is quite challenging and interesting. First approach used in achieving this goal of multi-threaded algorithm was to analyze the steps of the general algorithm to identify the steps that can be executed in parallel. Further analysis was focused on identifying the steps that can be interchanged without affecting the final result. Results obtained from the analysis were used to introduce the best way of implementing the multi core version of the algorithm. In implementing this, the java executor framework was used.

- **Efficient Java Programming Techniques**

In Java arrays differ from generic types in two important ways. First, arrays are covariant, which means simply that if Sub is a subtype of Super, then the array type Sub[] is a subtype of Super[]. Generics, by contrast, are invariant: for any two distinct types Type1 and Type2, List<Type1> is neither a subtype nor a supertype of List<Type2>. Also arguably arrays are deficient compared to generics.[8]

These Java language specific details encouraged the usage of Java Lists instead of arrays in every possible occasion in the implementation. Further, the type safeness in generics is used to avoid runtime exceptions which can occur in arrays due to type confusion.

6. Performance Results

The results presented in this section is based on the following testing environment.

Processor	- Intel Core 2 Duo 1.83GHz
Memory	- 2GB
Operating System	- Windows XP

The results presented here are focused only on the solving of the standard LP problem provided by the LP Problem Formulator explained in section 4.

6.1 Tranformer Performance

The standard LP problem generated has to be transformed into the Karmarkar's Canonical form in order to solve it by the solver. Therefore overall performance of the solving of the LP problem is dependent on the performance of the transformer. In this implementation, as mentioned in section 5, transformer is has two versions of implementations, the single core and the dual core version. The version, which will be running is dependent on the environment. Table 1 contains test results for the transformer.

Packages	Single Core(ms)	Dual Core(ms)
100	71	63
250	107	91
500	222	153
1000	1136	1008
1500	2832	2224
2000	5773	5058
2500	9988	8602
3000	32797	24121

Table 1 - Tranformer Performance

Graphed results are presented in figure 3

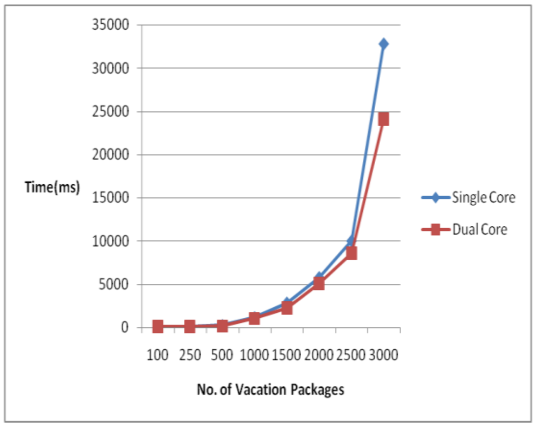

Figure 2 Tranformer Performance

6.2 Overall Solving Performance

Table 2, represents the overall performance of the LP problem solver including the transforming time.

No. of Packages	Solving Time(s)
25	5
50	12
75	24
100	41
125	61
150	89
175	112
200	155
225	210
250	290

Table 2 - Solver Performance

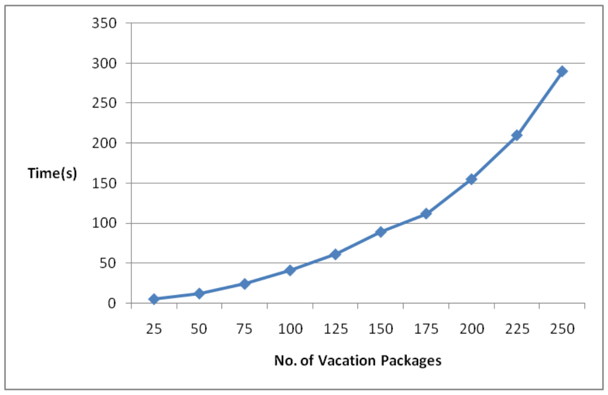

Figure 3 Solver Performance

7. Conclusion

By this it is evident that, Karmarkar's Projective Scaling Algorithm can be efficiently implemented in a programmatic way to solve real world large scale linear programming problems.

The Transformer and the Solver of this implementation in isolation can be easily used as linear programming problem solver due to the modularized implementation.

In this problem instance for the tour operator, the product planning cost can be lessoned, in the mean time being able to produce highly profitable set of vacation packages, so that the vacation package synthesis process will optimized ensuring high profits to the tour operator.

8. Acknowledgements

The authors wish to acknowledge the great guidance and support given by Dr. Shehan Perera, Mr. Yohan Welikala and Mr. Kishan Wimalarne for the successful completion of this effort.

9. References

[1] N.Karmarkar "A New Polynomial-time Algorithm for Linear Programming", Combrinatorica, 4, pp. 373-395, 1984

[2] N.Karmarkar and K.G. Ramakrishnan, "Implementation and Computational Results of the Karmarkar Algorithm for Linear Programming, Using an Iterative Method for Computing Projections",13th International Math. Prog. Symposium, Tokyo, Japan, August 1988.

[3] Edwin K.P. Chong, Stanislaw H. Zak, "An Introduction to Optimization", A Wiley-lnterscience Publication 2001

[4] Khachiyan, L. G., "A Polynomial Algorithm for Linear Programming," *Doklady Akad.Nauk USSR* **244,** 1093–1096, 1979, pages 1093–1096, Translated in *Soviet Math. Doklady* **20,** 1979, pages 191–194

[5] S. Wright: Interior - Point Methods, Computer Sciences Department, University of Wisconsin - Madison, 2002.

[6] David G. Luenburger, Yinyu Ye, "Linear and Nonlinear Programming-Third Edition", Springer Publication 2007

[7] "Numerical Methods for the STEM Undergraduate," [Online]. Available: http://numericalmethods.eng.usf.edu/topics/lu_decomposition.html [Accessed: July 7 2008].

[8] Joshua Bloch, "Effective Java-Second Edition", Addison-Wesley Publication 2001