Archives of Current Research International
3(2): 1-12, 2016, Article no.ACRI.23169
ISSN: 2454-7077

SCIENCEDOMAIN *international*
www.sciencedomain.org

Application of Linear Integer Programming to Optimal Resources Allocation: A Case Study of Mampong Municipal Assembly, Mampong-Ashanti

Nelson Opoku-Mensah[1], W. Obeng-Denteh[1*], Isaac Owusu-Mensah[2] and Frimpong Wiafe[3]

[1]Department of Mathematics, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana.
[2]Department of Science Education, University of Education, Winneba, Mampong-Ashanti, Ghana.
[3]Department of Religious Studies, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana.

Authors' contributions

This work was carried out in collaboration between all authors. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/ACRI/2016/23169
Editor(s):
(1) Yanpeng Li, Research Fellow, Department of Health Science Research, Mayo Clinic, USA.
Reviewers:
(1) Ette Harrison Etuk, Rivers State University of Science and Technology, Nigeria.
(2) Grienggrai Rajchakit, Maejo University, Thailand.
Complete Peer review History: http://sciencedomain.org/review-history/13147

Case Study

Received 17th November 2015
Accepted 18th December 2015
Published 3rd February 2016

ABSTRACT

The problem of allocating resources at the Mampong Municipal Assembly, Mampong-Ashanti with the aim of minimizing unnecessary lapses during budget allocation for resources by the assembly was considered. The problem was formulated as an Integer Linear Programming (ILP) problem using the available data from the Municipal Assembly. It was found that out of the twelve different locations considered and budget of Eight Hundred and Seventy Thousand Ghana Cedis, the optimal number of classroom to be built was thirty three (33) representing a three 3-unit classroom and four 6-unit classroom buildings at seven different locations within the Municipal at a minimum budget of Eight Hundred and Seven Thousand Ghana Cedis (GH¢ 807,000) respectively. We concluded that the Knapsack problem for selecting required sites in critical situations such as construction of school buildings was useful and it can be applied to any situation where allocation of

**Corresponding author: Email: obengdentehw@yahoo.com;*

funds in the sector of educational development becomes a serious setback. All of this will be achieved by using software called quantitative management which helps in solving and analyzing such problems.

Keywords: Integer linear programming; resources; optimal; school buildings; allocation.

1. INTRODUCTION

It is a universal truth that education is key to eradicating poverty in the modern society and this cannot be overemphasized. In a developing country like Ghana education will help citizens to acquire the needed skills and knowledge. The skill and knowledge acquired will make the citizens functionally literate and productive to facilitate poverty alleviation and promote the rapid socio-economic growth.

Education is a fundamental human right for all children and this right may not be realized in Ghana if strategic measures are not put in place to ensure adequate infrastructure provision to schools, especially in rural communities. School infrastructure is everything from electricity, toilets, safe buildings, tables, chairs, libraries, computer rooms, safe classrooms, sports fields, laboratories for science experiments, running water and fencing.

It is vital when we consider the fact that school infrastructure or resources, impact on how well teachers are able to teach and learners are able to learn. Learners attending schools with better infrastructure tend to perform better than learners who come from schools under trees. Meanwhile, the poor state of school infrastructure was evident in the number of public schools under trees. It is expected that all stakeholders particularly Civil Society, Government, Municipal, District and Metropolitans Assemblies ensure that funds provided are put into proper use.

The government allocated funds for putting up unit classrooms in these newly created Municipals. This calls for a scientific way or method that will help in the allocation of the provided fund in the Municipal in putting up schools in the community. The Municipal has allocated some fund to build unit classroom and must decide on which of these communities to put up the structures. Modern society, with advanced technology usually needs to make best possible decisions, which example involve the best possible use of resources or funds allocated to the educational sector to minimize production or guarantee full benefit of all.

Integer programs are beneficial because, if one can solve them, then one is guaranteed to obtain the best solution. However, this guarantee of optimality has a computational tradeoff, and integer programs currently may require exponential times to solve. The computational problems are so extreme that many integer programs cannot be solved, even using supercomputers [1].

The knapsack problem has been studied for more than a century, with early works dating as far back as 1897 for the reason that their direct application to problems arises in industries and also for their contribution to the solution methods for integer programming problems. Quite a lot of exact algorithms based on branch and bound, dynamic programming and heuristics have been proposed to solve the Knapsack Problems.

Renata and Grazia [2] offered an exact approach based on the optimal solution of sub-problems limited to a subset of variables. Each sub-problem is faced through a recursive variable-fixing process that continues until the number of variables decreases below a given threshold (restricted core problem). The solution space of the restricted core problem is split into subspaces, each containing solutions of a given cardinality. Each subspace is then explored with a branch-and-bound algorithm. Pruning conditions are introduced to improve the efficiency of the branch-and-bound routine.

The purpose of this paper is to model a real-life problem in developing sites for unit classroom at Mampong Municipal as a 0-1 knapsack problem, and propose branch-and-bound algorithm and to determine the maximum number of unit classrooms required to be built on selected sites and also results of the findings will be analyzed and interpreted.

The Mampong Municipal with Mampong as its capital is geographically located on the northern part of the region and shares boundary with Atebubu, Sekyere East, AfigyaSekyere and Ejura-Sekyere Dumasi to the north, east, south and west respectively. The Municipal is located within longitude 0.05° and 1.30° W and latitudes

6.55° and 7.30°N covering a total land area of 2346 km^2 in the Ashanti Region of Ghana. It has about 220 settlements with about 70% being rural. The rural areas are mostly found in the Afram Plains portion of the Municipal where Communities with less than fifty (50) people are scattered here and there.

The Municipal is generally low lying and gradually rising through rolling hills stretching southward towards Mampong. The highest point is 2400 m whilst the lowest is 135 m above mean sea level. It is fairly drained by several streams and rivers like Afram, Sene, Sasebonso, and Kyirimfa.

2. MATERIALS AND METHODS

The mathematical model will be formulated using linear programming based on the data provided at the budget office of the Mampong Municipal Assembly. The computer software package that will be used to solve and analyze the data is the Quantitative Management (QM software).

2.1 The Model

Michel**,** Perrot and Vanderbeck [3] considered the multiple-class integer knapsack problem with setups. Items are partitioned into classes whose use imply a setup cost and associated capacity consumption. Item weights are assumed to be a multiple of their class weight. The total weight of selected items and setups is bounded. The objective is to maximize the difference between the profits of selected items and the fixed costs incurred for setting-up classes. The authors showed the extent to which classical results for the knapsack problem can be generalized to these variants with setups. In particular, an extension of the branch-and-bound algorithm of Horowitz and Sahni [4] is developed for problems with positive setup costs. Yan and Chen [5] developed a model that help Taiwanese intercity bus carriers in timetable settings and bus routing or scheduling. The model employs multiple time-space networks that can formulate bus movements and passenger flows and manage the interrelationships between passenger trip demands and bus trip suppliers to produce the best timetables and bus routes or schedules. Kosuch, Le Bodic & Lisser [6] studied the stochastic knapsack problem with expectation constraint. The item weights are assumed to be independently normally distributed. The authors solved the relaxed version of this problem using a stochastic gradient algorithm in order to provide upper bounds for a branch-and-bound framework. The 0-1 knapsack problem is a linear integer-programming problem with a single constraint and binary variables. The knapsack problem with an inequality constraint has been widely studied, and several efficient algorithms have been published. Balasubramanian and Sanjiv [7] considered the equality-constraint knapsack problem, which has received relatively little attention. The authors described a branch-and-bound algorithm for this problem, and present computational experience with up to 10,000 variables. An important feature of this algorithm is a least-lower-bound discipline for candidate problem selection.

2.2 Linear Programming

Linear programming is a mathematical method for determining a way to achieve the best outcome (such as maximum profit or lowest cost) in a given mathematical model for some list of requirements represented as linear relationships. Linear programming is a specific case of mathematical programming (mathematical optimization).

More formally, linear programming is a technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints. Its feasible region is a convex polyhedron, which is a set defined as the intersection of finitely many half spaces, each of which is defined by a linear inequality. Its objective function is a real-valued affine function defined on this polyhedron. A linear programming algorithm finds a point in the polyhedron where this function has the smallest (or largest) value if such a point exists.

Linear programs are problems that can be expressed in canonical form:

$$\begin{aligned} &\text{maximize } \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}$$

$\text{and } \mathbf{x} > 0$, where x represents the vector of variables (determined), c and b are vectors of (known) coefficients, *A* is a (known) matrix of coefficients, and $(.)^{\mathrm{T}}$ are the matrix transpose. The expression to be maximized or minimized is called the objective function ($c^{\mathrm{T}}x$ in this case). The inequalities $Ax \leq b$ is the constraints which specify a convex polytope over which the objective function is to be optimized. In this

context, two vectors are comparable when they have the same dimensions. If every entry in the first is less-than or equal-to the corresponding entry in the second then we can say the first vector is less-than or equal-to the second vector.

2.3 Complete Linear Programming Model

Combining the aforementioned components into a single statement gives:

$$maximize\ or\ minimize z = \sum_{j=1}^{n} c_j x_j$$

$$subject\ to \sum_{j=1}^{n} a_{ij} x_j \begin{Bmatrix} \leq \\ = \\ \geq \end{Bmatrix} b_i, for\ i = 1 \dots m$$

$$0 \leq x_j \leq u_j\ for\ j = 1 \dots n$$

The constraints, including non-negativity and simple upper bounds, define the feasible region of a problem.

2.4 Assumptions of Linear Programming

For a problem to be realistically represented as a linear program, the following assumptions should hold:

(i) The constraints and objective function are linear.

(a) This requires that the value of the objective function and the response of each resource expressed by the constraints are proportional to the level of each activity expressed in the variables.

(b) Linearity also requires that the effects of the value of each variable on the values of the objective function and the constraints are additive. In other words, there can be no interactions between the effects of different activities; i.e., the level of activity X_1 should not affect the costs or benefits associated with the level of activity X_2.

(ii) Divisibility: The values of decision variables can be fractions. Sometimes these values only make sense if they are integers; then we need an extension of linear programming called integer programming.

(iii) Certainty: The model assumes that the responses to the values of the variables are exactly equal to the responses represented by the coefficients.

(iv) Data: Formulating a linear program to solve a problem assumes that data are available to specify the problem.

2.5 Knapsack Problem

The knapsack problem is one of the most studied problems in combinatorial optimization, with many real-life applications. For this reason, many special cases and generalizations have been examined.

Common to all versions are a set of n items, with each item $1 \leq j \leq n$ having an associated profit p_j and weigh tw_j. The objective is to pick some of the items, with maximal total profit, while obeying that the maximum total weight of the chosen items must not exceed W. Generally, these coefficients are scaled to become integers, and they are almost always assumed to be positive.

The knapsack problem in its most basic form:

$$maximize \sum_{j=1}^{n} p_j x_j$$

$$subjectto \sum_{i=1}^{n} w_j x_j \leq W, \qquad x_j \in \{0,1\} \forall j \in \{1, \dots, n\}$$

Knapsack problems appear in real-world decision-making processes in a wide variety of fields, such as finding the least wasteful way to cut raw materials, selection of capital investments and financial portfolios, selection of assets for asset-backed securitization, and generating keys for the cryptosystem. One early application of knapsack algorithms was in the construction and scoring of tests in which the test-takers have a choice as to which questions they answer. For small examples it is a fairly simple process to provide the test-takers with such a choice. For example, if an exam contains 12 questions each worth 10 points, the test-taker need only answer 10 questions to achieve a maximum possible score of 100 points. However, on tests with a heterogeneous distribution of point values, that is, different questions are worth different point values, it is more difficult to provide choices. Feuerman and Weiss [8] proposed a system in which students are given a

heterogeneous test with a total of 125 possible points. The students are asked to answer all of the questions to the best of their abilities. Of the possible subsets of problems whose total point values add up to 100, a knapsack algorithm would determine which subset gives each student the highest possible score.

2.6 0-1 Knapsack Problem

The most common problem being solved is the 0-1 knapsack problem, which restricts the number x_i of copies of each kind of item to zero or one.

Mathematically the 0-1-knapsack problem can be formulated as:

Let there be n items, x_1 to x_n where x_i has a value v_i and weight w_i. The maximum weight that the bag can carry is W. It is common to assume that all values and weights are nonnegative. To simplify the representation, it is assumed that the items are listed in increasing order of weight.

$$maximize \sum_{i=1}^{n} v_i x_i$$

$$subject\ to \sum_{i=1}^{n} w_i x_i \leq W, x_i \in \{0,1\}$$

Maximize the sum of the values of the items in the knapsack so that the sum of the weights must be less than the knapsack's capacity.

A similar dynamic programming solution for the 0-1 knapsack problem also runs in pseudo-polynomial time. Assume $w_1, w_2, \ldots, w_n, W$ are strictly positive integers. Define $m[i, w]$ to be the maximum value that can be attained with weight less than or equal to w using items up to i.

Thus $m[i, w]$ can be defined recursively as follows:

$(i) m[i, w] = m[i-1, w]$ if $w_i > w$ (the new item is greater than the existing weight limit).

$(ii) m[i, w] = \max(m[i-1, w], m[i-1, w - w_i] + v_i)$ if $w_i \leq w$. .

The solution can then be found by calculating $m[n, W]$. To do this efficiently we can use a table to store preceding computations.

2.7 Branch and Bound

The basic concept underlying the branch-and-bound technique is to divide and conquer. The process contains dividing (branching) original large problem into smaller sub problems and bounding the best solution in the subsets.

The steps are;

(i) Solve the problem without integer restrictions,
(ii) If the solution is integer , then this must be the solution to integer problem,
(iii) If these variables are not integer valued, the feasible region is divided by adding constraints restricting the value of one of the variables that was not integer valued,
(iv) Bounds on the value of the objective function are found and used to help determine which sub-problems can be eliminated and when the optimal solution has been found,
(v) If a solution is not optimal, a new sub-problem is selected and branching continues.

Branch and bound (BB or B&B) is a general algorithm for finding optimal solutions of various optimization problems, especially in discrete and combinatorial optimization. A Branch-and-Bound algorithm consists of a systematic enumeration of all admissible solutions, where large subsets of fruitless candidates are discarded en masse, by using upper and lower estimated bounds of the quantity being optimized.

3. DATA COLLECTION AND ANALYSIS

In an effort to develop the educational infrastructure in the Municipal, the Assembly proposed a budget of $GH¢870{,}000.00$ for the development of lands and the construction of Unit classroom buildings. These buildings are to be constructed at twelve different towns within the Municipal, whose estimated capacities in terms of the number of unit classrooms and development cost are given in Table 1. The respective towns to be considered within the Municipal are Kofiase, Kyekyewere, Asaam, Benim, Atonsuagya, Yonso, Apaah, Adidwan, Nyinampong, Abuontem, Nkwanta and Penteng.

A unit classroom is made up of the number of study rooms, an office, a store, staff common room and a toilet facility. The type of unit

classroom to be built in each town was based on the population, existing educational infrastructures. All three unit classrooms are for JHS, six unit classroom is for the lower and upper primary schools, and the nine unit classroom is for both primary and JHS. The difference in the cost for the same unit classroom was due to different construction works to be done on the various lands. Appendix 1 provides the breakdown of the budget of associated cost for each unit classroom for the various locations.

Table 1. Respective towns with their budget allocations

Towns	Capacity (unit classroom)	Cost (GH¢ 1000)
Kofiase	3	83
Kyekyewere	6	169
Asaam	9	270
Benim	3	75
Atonsuagya	3	103
Yonso	6	145
Apaah	3	97
Adidwan	6	139
Nyinampong	3	73
Abuontem	6	149
Nkwanta	9	267
Penteng	6	143

The dilemma here is to choose suitable locations in such a way that the optimal capacity would be attained without exceeding the budget allocated for project.

With a link to the Knapsack Problem model, the holding capacity of the resource maximum value is the Assembly's budget. The various items to be measured are the different sites (lands) that can be developed for the project, the weight of any item is the cost of developing and construction of the project and the value of each item is the capacity of each site.

The problem can therefore be modeled as:

$$Maximize \quad \sum_{i=1}^{n} c_i x_i$$

$$Subject\ to \sum_{i=1}^{n} w_i x_i \leq W, \quad x_i \in \{0,1\}, \quad i = 1, \ldots\ldots, n$$

Where;

C = Total capacity
c_i = Capacity of each item or site
x_i = Number of sites developed
w_i = Cost of developing a site
W = Total budget for the development (resource limit)

Thus,

$$maximize\ C = 3X_1 + 6X_2 + 9X_3 + 3X_4 + 3X_5 + 6X_6 + 3X_7 + 6X_8 + 3X_9 + 6X_{10} + 9X_{11} + 6X_{12}$$

$$subject\ to\ 83X_1 + 169X_2 + 270X_3 + 75X_4 + 103X_5 + 145X_6 + 97X_7 + 139X_8 + 73X_9 + 149X_{10} + 267X_{11} + 143X_{12} \leq 870$$

A Branch and Bound algorithm model is applied to carry out the computation of the model. The items to be considered are twelve (which means $n = 12$) consisting of Kofiase, Kyekyewere, Asaam, Benim, Atonsuagya, Yonso, Apaah, Adidwan, Nyinampong, Abuontem, Nkwanta and Penteng.

The weights of each item are $w_1 = 83, w_2 = 169, w_3 = 270, w_4 = 75, w_5 = 103, w_6 = 145, w_7 = 97, w_8 = 139, w_9 = 73, w_{10} = 149, w_{11} = 267, w_{12} = 143$ whiles the values of each item are $X_1 = 3, X_2 = 6, X_3 = 9, X_4 = 3, X_5 = 3, X_6 = 6, X_7 = 3, X_8 = 6, X_9 = 3, X_{10} = 6, X_{11} = 9, X_{12} = 6$ and the maximum available budget fund $W = 870$.

Note: For locations

X1: Kofiase X7: Apaah
X2: Kyekyewere X8: Adidwan
X3: Asaam X9: Nyinampong
X4: Benim X10: Abuontem
X5: Atonsuagya X11: Nkwanta
X6: Yonso X12: Penteng

4. RESULTS OF THE ANALYSIS

Results of the analysis in obtaining maximum number of unit classroom buildings at selected location in the Municipal are shown in the tables below. The tables provide a breakdown of the associated cost in building the unit classroom. The optimal selection of unit classrooms yielded eight hundred and seven thousand Ghana Cedis (GH¢807,000). The amount is able to construct a three 3-unit classroom building at Kofiase, Benim and Nyinampong and four 6-unit classroom building at Yonso, Adidwan, Abuontem and Penteng respectively. Thus the total number of classroom to be built out of budget is 33. This means that out of the total budget of Eight Hundred and Seventy Thousand Ghana Cedis (GH¢ 870,000) which was proposed by the Assembly, an excess amount of Sixty Three Thousand Ghana Cedis (GH¢ 63,000) was left.

This excess amount can be used to undertake other project in the Municipal.

4.1 Sensitivity Analysis on the Whole Solution

The Sensitivity Analysis is often used for integer linear programming problem than the Linear Programming (LP) problem. That is, a very small change in one of the coefficients in the constraints can cause a reasonably large change in the optimal value. In the case of our study, any time there is a change in any of the amount of the budget allocation, and then the integer linear program problem has to be resolved with slight variation in the coefficients before an optimal solution is chosen for implementation.

5. CONCLUSION

The research sought to use the Knapsack problem for selecting required sites in critical situations such as construction of school buildings. However, it can be applied to any situation where allocation of funds in the sector of development becomes a serious problem. A minimum amount of eight hundred and seven thousand Ghana cedis (GH¢807,000) was obtained in construction of a three 3-unit and four 6-unit classroom buildings at seven different locations within the Municipal to enhance the educational development.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Dahl G. An introduction to convexity, polyhedral theory and combinatorial optimization. University of Oslo, Department of Informatics; 1997.
2. Mansini R, Speranza MG. An exact algorithm for the multidimensional knapsack problem; 2009.
   Available:http://ideas.repec.org/a/eee/ejores/v196y2009i3p909-918
3. Michel S, Perrot N, Vanderbeck F. Knapsack problems with setups; 2009.
   Available:http://ieeexplore.ieee.org/xpl/freeabs
4. Horowitz E, Sahni S. Computing partitions with applications to knapsack problems. Journal of ACM. 1974;21:277-292.
5. Yan S, Chen HL. A scheduling model and a solution algorithm for inter-city bus carriers. Transportation Research Part A: Policy & Practice. 2002;36:805.
6. Kosuch S, Le Bodic P, Leung J, Lisser A. On Stochastic Bilevel Programming Problem with Knapsack Constraints; 2009.
   Available:http://www.kosuch.eu/stefanie/veroeffentlichungen
7. Balasubramanian R, Sanjiv S. An algorithm for the 0-1 equality knapsack problem. Journal of the Operational Research Society. 1988;39:1045–1049.
8. Available:https://en.wikipedia.org/wiki/Knapsack_problem

APPENDIX 1

Breakdown of budget allocations for various sites

Item	Description	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
A	Preliminaries	5,450	2,623	17,463	4,320	8,450	1,545	5,950	2,689	2,761	1,823	17,263	2,435
B	Excavation and Earthworks	2,2032	35,891	50,499	20,214	16,532	28,321	19,532	37,105	18,282	33,891	50,490	27,134
C	Concrete works	6,482	14,473	15,841	4,361	11,182	8,923	11,482	12,254	5,938	9,473	15,841	8,374
D	Block works	4,216	5,987	17,499	4,321	9,916	5,579	9,216	8,065	4,200	4,787	17,500	5,372
E	Roofing to summary	3,788	9,321	10,377	3,680	4,738	6,819	4,288	8,910	3,370	8,421	10,380	8,934
F	Carpentry works	9,160	8,880	10,044	7,670	9,660	6,621	9,360	8,790	6,423	7,820	10,040	10,009
G	Joinery/Walling	2,508	18,914	20,433	2,452	3,208	18,091	3,008	20,027	3,342	18,464	20,421	15,782
H	Metal works	8,531	8,927	18,844	7,462	9,831	6,892	9,031	610	6,443	5,927	18,717	7,280
I	Plastering work/floor	3,777	17,342	19,321	2,992	8,077	14,510	7,277	15,828	2,934	16,842	19,322	14,489
J	Painting/decoration	3,527	9,272	10,372	3,341	4,427	4,752	3,727	6,410	3,531	5,672	10,373	6,730
K	External works	5,112	5,973	12,213	4,670	6,312	3,989	5,512	6,800	4,520	5,423	12,213	6,645
L	Construction of ramps	1,587	1,939	9,852	1,382	2,237	789	1,787	605	1,363	1,489	9,853	4,849
M	Electrical works	4,259	5,490	18,479	4,540	6,359	5,258	4,359	8,512	4,950	5,440	17,847	5,344
N	Surplus Amount	2,571	24,028	38,763	3,595	2,071	32,911	2,471	2,395	4,943	23,528	36,740	19,623
	Total	**83,000**	**169,000**	**270,000**	**75,000**	**103,000**	**145,000**	**97,000**	**139,000**	**73,000**	**149,000**	**267,000**	**143,000**

APPENDIX 2

Optimal solutions for the various iterative stages (output from QM software)

Iteration	Level	Added constraint	Solution type	Solution value	X 1	X 2	X 3	X 4	X 5	X 6	X 7	X 8	X 9	X 10	X 11	X 12	Cost
			Optimal	33	1	0	0	1	0	1	0	1	1	1	0	1	807
1	5	X5<= 0	Integer	33	1	0	0	1	0	1	0	1	1	1	0	1	807
2	6	X1<= 0	Integer	33	0	0	0	1	1	1	0	1	1	1	0	1	827
3	7	X4<= 0	Integer	33	1	0	0	0	1	1	0	1	1	1	0	1	835
4	8	X10<= 0	Sub optimal	30	1	0	0	1	1	1	0	1	1	0	0	1	761
5	9	X9<= 0	Integer	33	1	0	0	1	1	1	0	1	0	1	0	1	837
6	10	X6<= 0	Sub optimal	30	1	0	0	1	1	0	0	1	1	1	0	1	765
7	11	X12<= 0	Sub optimal	30	1	0	0	1	1	1	0	1	1	1	0	0	767
8	12	X8<= 0	Sub optimal	30	1	0	0	1	1	1	0	0	1	1	0	1	771
9	6	X5<= 0	Integer	33	0	0	0	1	0	1	1	1	1	1	0	1	821
10	7	X4<= 0	Integer	33	0	0	0	0	1	1	1	1	1	1	0	1	849
11	8	X10<= 0	Sub optimal	30	0	0	0	1	1	1	1	1	1	0	0	1	775
12	9	X9<= 0	Integer	33	0	0	0	1	1	1	1	1	0	1	0	1	851
13	10	X6<= 0	Sub optimal	30	0	0	0	1	1	0	1	1	1	1	0	1	779
14	11	X12<= 0	Sub optimal	30	0	0	0	1	1	1	1	1	1	1	0	0	781
15	12	X8<= 0	Sub optimal	30	0	0	0	1	1	1	1	0	1	1	0	1	783
16	7	X5<= 0	Integer	33	1	0	0	0	0	1	1	1	1	1	0	1	829
17	8	X10<= 0	Sub optimal	30	1	0	0	0	1	1	1	1	1	0	0	1	783
18	9	X9<= 0	Integer	33	1	0	0	0	1	1	1	1	0	1	0	1	859
19	10	X6<= 0	Sub optimal	30	1	0	0	0	1	0	1	1	1	1	0	1	787
20	11	X12<= 0	Sub optimal	30	1	0	0	0	1	1	1	1	1	1	0	0	789
21	12	X8<= 0	Sub optimal	30	1	0	0	0	1	1	1	0	1	1	0	1	793
22	7	X10<= 0	Integer	33	1	0	0	1	1	1	1	1	1	0	0	1	858
23	9	X5<= 0	Integer	33	1	0	0	1	0	1	1	1	0	1	0	1	835
24	10	X6<= 0	Sub optimal	30	1	0	0	1	1	0	1	1	0	1	0	1	789
25	11	X12<= 0	Sub optimal	30	1	0	0	1	1	1	1	1	0	1	0	0	791
26	12	X8<= 0	Sub optimal	30	1	0	0	1	1	1	1	0	0	1	0	1	795
27	9	X6<= 0	Integer	33	1	0	0	1	1	0	1	1	1	1	0	1	862

Iteration	Level	Added constraint	Solution type	Solution value	X 1	X 2	X 3	X 4	X 5	X 6	X 7	X 8	X 9	X 10	X 11	X 12	Cost
28	10	X12<= 0	Integer	33	1	0	0	1	1	1	1	1	1	1	0	0	864
29	11	X8<= 0	Integer	33	1	0	0	1	1	1	1	0	1	1	0	1	868
30	7	X5<= 0	Integer	33	0	0	1	1	0	1	0	1	1	0	0	1	845
31	8	X5<= 0	Integer	33	0	0	1	0	0	1	1	1	1	0	0	1	867
32	9	X5<= 0	Integer	33	0	0	1	1	0	1	1	1	0	0	0	1	869
33	8	X5<= 0	Integer	33	1	0	1	0	0	1	0	1	1	0	0	1	853
34	9	X5<= 0	Integer	33	1	0	1	1	0	1	0	1	0	0	0	1	855
35	9	X5<= 0	Integer	33	0	0	1	0	0	1	0	1	0	1	0	1	846
36	9	X5<= 0	Integer	33	1	0	1	1	0	0	0	1	0	1	0	1	859
37	10	X5<= 0	Integer	33	1	0	1	1	0	1	0	1	0	1	0	0	861
38	11	X5<= 0	Integer	33	1	0	1	1	0	1	0	0	0	1	0	1	865
39	9	X5<= 0	Integer	33	0	0	1	1	0	0	0	1	1	1	0	1	849
40	10	X5<= 0	Integer	33	1	0	1	0	0	0	0	1	1	1	0	1	857
41	10	X5<= 0	Integer	33	0	0	1	1	0	1	0	1	1	1	0	0	851
42	11	X5<= 0	Integer	33	1	0	1	0	0	1	0	1	1	1	0	0	859
43	11	X5<= 0	Integer	33	0	0	1	1	0	1	0	0	1	1	0	1	855
44	12	X5<= 0	Integer	33	1	0	1	0	0	1	0	0	1	1	0	1	863
45	7	X5<= 0	Integer	33	0	0	0	1	0	1	0	1	1	0	1	1	842
46	7	X5>= 1	Integer	33	0	0	0	0	1	1	0	1	1	0	1	1	870
47	8	X5<= 0	Integer	33	0	0	0	0	0	1	1	1	1	0	1	1	864
48	9	X5<= 0	Integer	33	0	0	0	1	0	1	1	1	0	0	1	1	866
49	8	X5<= 0	Integer	33	1	0	0	0	0	1	0	1	1	0	1	1	850
50	9	X5<= 0	Integer	33	1	0	0	1	0	1	0	1	0	0	1	1	852
51	9	X5<= 0	Integer	33	0	0	0	0	0	1	0	1	0	1	1	1	843
52	9	X5<= 0	Integer	33	1	0	0	1	0	0	0	1	0	1	1	1	856
53	8	X7<= 0	Integer	33	0	0	0	1	0	0	1	1	0	1	1	1	870
54	10	X5<= 0	Integer	33	1	0	0	1	0	1	0	1	0	1	1	0	858
55	11	X5<= 0	Integer	33	1	0	0	1	0	1	0	0	0	1	1	1	862
56	9	X5<= 0	Integer	33	0	0	0	1	0	0	0	1	1	1	1	1	846
57	10	X5<= 0	Integer	33	0	0	0	0	0	0	1	1	1	1	1	1	868
58	10	X5<= 0	Integer	33	1	0	0	0	0	0	0	1	1	1	1	1	854
59	10	X5<= 0	Integer	33	0	0	0	1	0	1	0	1	1	1	1	0	848
60	9	X7>= 1	Integer	33	0	0	0	0	0	1	1	1	1	1	1	0	870

Iteration	Level	Added constraint	Solution type	Solution value	X 1	X 2	X 3	X 4	X 5	X 6	X 7	X 8	X 9	X 10	X 11	X 12	Cost
61	11	X5<= 0	Integer	33	1	0	0	0	0	1	0	1	1	1	1	0	856
62	11	X5<= 0	Integer	33	0	0	0	1	0	1	0	0	1	1	1	1	852
63	12	X5<= 0	Integer	33	1	0	0	0	0	1	0	0	1	1	1	1	860
64	7	X5<= 0	Integer	33	0	1	0	0	0	1	0	1	1	1	0	1	818
65	9	X9<= 0	Integer	33	0	1	0	0	1	1	0	1	0	1	0	1	848
66	7	X10<= 0	Integer	33	0	1	0	0	1	1	1	1	1	0	0	1	869
67	9	X5<= 0	Integer	33	0	1	0	0	0	1	1	1	0	1	0	1	842
68	9	X5<= 0	Integer	33	0	1	1	0	0	1	0	1	0	0	0	1	866
69	7	X10>= 1	Integer	33	0	1	1	0	0	0	0	1	0	1	0	1	870
70	9	X5<= 0	Integer	33	0	1	0	0	0	1	0	1	0	0	1	1	863
71	10	X5<= 0	Integer	33	0	1	0	0	0	0	0	1	0	1	1	1	867
72	11	X5<= 0	Integer	33	0	1	0	0	0	1	0	1	0	1	1	0	869
73	7	X5<= 0	Integer	33	1	1	0	0	0	1	1	1	1	0	0	1	849
74	8	X7<= 0	Integer	33	1	1	0	0	1	1	0	1	1	0	0	1	855
75	9	X5<= 0	Integer	33	1	1	0	0	0	1	0	1	0	1	0	1	828
76	9	X5<= 0	Integer	33	1	1	0	0	0	0	1	1	1	1	0	1	853
77	10	X7<= 0	Integer	33	1	1	0	0	1	0	0	1	1	1	0	1	859
78	10	X5<= 0	Integer	33	1	1	0	0	0	1	1	1	1	1	0	0	855
79	11	X7<= 0	Integer	33	1	1	0	0	1	1	0	1	1	1	0	0	865
80	11	X5<= 0	Integer	33	1	1	0	0	0	1	1	0	1	1	0	1	859
81	12	X7<= 0	Integer	33	1	1	0	0	1	1	0	0	1	1	0	1	865
82	7	X5<= 0	Integer	33	1	1	0	1	0	1	0	1	1	0	0	1	827
83	8	X1<= 0	Integer	33	0	1	0	1	1	1	0	1	1	0	0	1	847
84	9	X9<= 0	Integer	33	1	1	0	1	1	1	0	1	0	0	0	1	857
85	10	X6<= 0	Sub optimal	30	1	1	0	1	1	0	0	1	1	0	0	1	785
86	11	X12<= 0	Sub optimal	30	1	1	0	1	1	1	0	1	1	0	0	0	787
87	12	X8<= 0	Sub optimal	30	1	1	0	1	1	1	0	0	1	0	0	1	791
88	8	X5<= 0	Integer	33	0	1	0	1	0	1	1	1	1	0	0	1	841
89	9	X5<= 0	Integer	33	1	1	0	1	0	1	1	1	0	0	0	1	851
90	9	X5<= 0	Integer	33	0	1	1	1	0	0	0	1	1	0	0	1	869
91	9	X5<= 0	Integer	33	0	1	0	1	0	0	0	1	1	0	1	1	866
92	10	X5<= 0	Integer	33	0	1	0	1	0	1	0	1	1	0	1	0	868
93	9	X5<= 0	Integer	33	0	1	0	1	0	1	0	1	0	1	0	1	820

Iteration	Level	Added constraint	Solution type	Solution value	X 1	X 2	X 3	X 4	X 5	X 6	X 7	X 8	X 9	X 10	X 11	X 12	Cost
94	10	X6<= 0	Sub optimal	30	0	1	0	1	1	0	0	1	0	1	0	1	778
95	11	X12<= 0	Sub optimal	30	0	1	0	1	1	1	0	1	0	1	0	0	780
96	12	X8<= 0	Sub optimal	30	0	1	0	1	1	1	0	0	0	1	0	1	784
97	9	X5<= 0	Integer	33	1	1	0	1	0	0	1	1	0	1	0	1	855
98	10	X7<= 0	Integer	33	1	1	0	1	1	0	0	1	0	1	0	1	861
99	10	X5<= 0	Integer	33	1	1	0	1	0	1	1	1	0	1	0	0	857
100	11	X7<= 0	Integer	33	1	1	0	1	1	1	0	1	0	1	0	0	863
101	11	X5<= 0	Integer	33	1	1	0	1	0	1	1	0	0	1	0	1	861
102	12	X7<= 0	Integer	33	1	1	0	1	1	1	0	0	0	1	0	1	867
103	9	X5<= 0	Integer	33	1	1	0	1	0	0	0	1	1	1	0	1	831
104	10	X1<= 0	Integer	33	0	1	0	1	1	0	0	1	1	1	0	1	851
105	11	X12<= 0	Sub optimal	30	1	1	0	1	1	0	0	1	1	1	0	0	791
106	12	X8<= 0	Sub optimal	30	1	1	0	1	1	0	0	0	1	1	0	1	795
107	10	X5<= 0	Integer	33	0	1	0	1	0	0	1	1	1	1	0	1	845
108	10	X5<= 0	Integer	33	1	1	0	1	0	1	0	1	1	1	0	0	833
109	11	X1<= 0	Integer	33	0	1	0	1	1	1	0	1	1	1	0	0	853
110	12	X8<= 0	Sub optimal	30	1	1	0	1	1	1	0	0	1	1	0	0	797
111	11	X5<= 0	Integer	33	0	1	0	1	0	1	1	1	1	1	0	0	847
112	11	X5<= 0	Integer	33	1	1	0	1	0	1	0	0	1	1	0	1	837
113	12	X1<= 0	Integer	33	0	1	0	1	1	1	0	0	1	1	0	1	857
114	12	X5<= 0	Integer	33	0	1	0	1	0	1	1	0	1	1	0	1	851

Peer-review history:
The peer review history for this paper can be accessed here:
http://sciencedomain.org/review-history/13147