Recursive MILP model to find alternate solutions in Linear Programming models

Dev A. Kakkad[a], Ignacio E. Grossmann[a], Bianca Springub[b], Christos Galanopoulos[b] Leonardo Salsano de Assis[b], & Nga Tran[b]

[a] *Department of Chemical Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA*
[b] *Aurubis AG, Hamburg 20539, Germany*

Abstract

We address in this paper LP models in which it is desired to find alternate optima. An LP may have multiple alternate solutions with the same objective value or with increasing objective values. To obtain a specified number of these alternate solutions in the increasing order of objective values, we propose a recursive MILP algorithm. We demonstrate the application and effectiveness of this algorithm on a 2 dimensional LP and on small and large supply chain problems.

Keywords: Supply Chain network, Linear Programming, Mixed Integer linear programming, Alternate Optima

1 Introduction

Linear programming (LP) has found numerous applications across various domains due to its ability to optimize limited resources and make informed decisions. In operations research and supply chain management, LP has been utilized for production planning, inventory management, and distribution optimization (Maravelias et al., 2009). LP has also played a vital role in transportation and logistics, enabling efficient routing and scheduling of vehicles (Yang et al., 2015). In finance, LP has been employed for portfolio optimization, asset allocation, and risk management (Luenberger and Ye, 2008). Additionally, LP has been applied in energy systems, healthcare resource allocation, telecommunications network optimization, and many other fields (Gass S.I., 2003). Its versatility and effectiveness make linear programming a valuable tool for decision-making in complex real-world problems.

Supply chain optimization has become increasingly important for businesses to ensure flexibility and responsiveness to changes in the market. A variety of LP, MILP, and MINLP models have been used in the industry to deal with strategic and operational problems associated with the design, long-term/midterm planning, and short-term operations of supply chains (Grossmann, 2005; Harjunkoski et al., 2014). Although these models are useful, some challenges may arise when they are implemented in operation. In large scale supply chain models, interactions between the different nodes of the supply chain model lead to a complex structure wherein making any changes to parameters can impact the output of the entire model (Cafaro and Grossmann, 2014; You and Grossmann, 2008). This leads to the decision-making process being a challenging task when changes are introduced to the model and drive the need for identifying alternate solutions of the LP model. Furthermore, these models can be used by the operators at the shop-floor level to carry out their day to day operations, or it can be used by the operations planner in an enterprise to understand and interpret their supply chain.

In biological systems, degeneracy in linear programming models for metabolic networks represents multiple pathways and reaction networks that perform similar functions. In biological and chemical reactive systems, this phenomenon is often recognized and used to find different pathways to produce the same product. It is helpful when it is physically challenging to achieve certain optimal pathways. For identifying alternative optimum solutions in linear programming models, Lee et al. (2000) introduced a recursive MILP approach and used it to study metabolic networks. In this paper we propose an improved recursive MILP approach that will not only obtain alternate optimal solutions but will produce a solution pool of all the alternate solutions in the increasing order of value of the objective function.

Alternate solutions can also be interpreted as the extreme points that lie at the intersection of the convex polytope given by the linear constraints and the hyperplane of the optimal objective function value. A number of special purpose algorithms have been reported in the literature for finding all the extreme points of the convex polytope of a Linear Programming problem (Swart, 1985; Matheiss and Rubin, 1980). Aside from the fact that these algorithms have exponential complexity, they are not easy to implement. Therefore, our goal in this study is to develop a recursive MILP method, which has the advantage that it can readily be implemented in a modeling language (e.g. GAMS, AMPL, Pyomo).

The paper is organized as follows. Section 2 discusses the motivation for a linear programming supply chain optimization problem. In section 3, the problem statement is given for finding alternate solutions for general LP problems. In section 4, this paper introduces a comprehensive novel algorithm to find multiple alternate solutions of a given LP problem. After presenting the general methodology, Section 5 demonstrate the method on a 2 dimensional Linear Programming problem and on both a small and large supply chain model provided by Aurubis AG. Section 6 draws conclusions for the paper.

2 Motivating Problem for finding alterante optima

Linear network flow problems are known to often have a high level of degeneracy. Therefore, they may exhibit multiple optimal solutions that have the same objective value. Finding all such solutions can provide useful information to the decision maker instead of simply reporting one single solution. Investigating the possibility of other solutions or the next-best solution in network flow problems offers even another layer of knowledge about the structure of the network problems. Providing the user of these models with a solution pool of N best solutions can provide useful information to the model user.

As a specific example of a linear network flow problem consider the graphical representation of a typical supply chain model shown in Figure 1. As the materials flow from the suppliers to the customers, the goal is to optimize the flow to ensure the optimal coordination between the different components of the supply chain. In many occasions, disruptions or changes in the model may be introduced, which require resolving the LP model. Rather than obtaining a single solution from the LP model as is common practice, we aim to identify a specified number of alternate best solutions of the LP model, which may include degenerate and no-degenerate solutions. In the next section we address this problem in general form

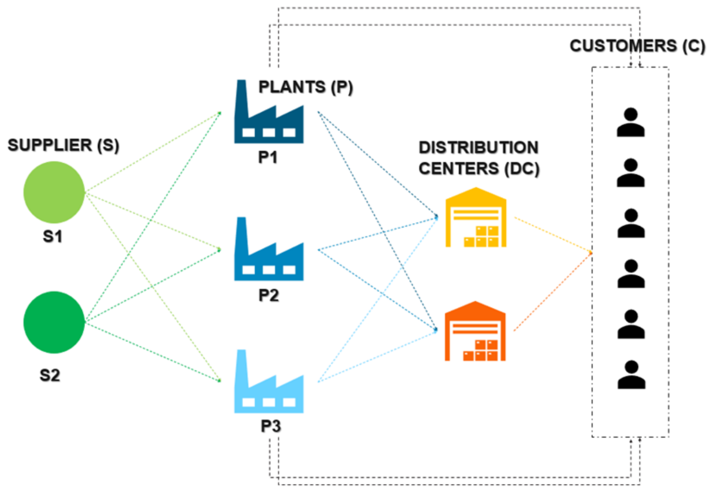


Figure 1: Supply Chain Layout

3 Problem Statement

Alternate Solutions of an LP - Given an LP model shown in problem (1) below, our goal is to find the optimal solution as well as alternate solutions with the same objective value or with increasing objective value. More specifically, given an LP model, the goal is to find the N best solutions

$$
\begin{aligned}
\min \quad & z = c^T x \\
s.t. \quad & Ax = B \\
& Dx \leq e \\
& x \geq 0
\end{aligned} \tag{1}
$$

We propose in the next section an algorithm that can provide the user with a solution pool of the N best solutions of the LP.

4 Algorithm to find alternate optimal solutions in linear programming models

4.1 Properties of an LP

Linear programming corresponds mathematically to the optimization of a linear objective function, subject to linear equality and linear inequality constraints (Charnes & Cooper, 1957). A linear programming problem in its general form is given by model (1) in Section 3. We rewrite the model and include the variable bounds in the set of inequality constraints - $Ex \leq d$. The resulting model is given by model (2)

$$
\begin{aligned}
\min \quad & z = c^T x \\
s.t. \quad & Ax = B \\
& Ex \leq d \\
& x \in R^n
\end{aligned} \tag{2}
$$

An important property of a Linear programming problem is that the optimal solution is found at the extreme points of the feasible space (Chvatal, 1983). Algorithms such as the simplex algorithm (Dantzig, 1982) and the interior point methods (Karmarkar, 1984) only find one extreme point. If we want to find alternative solutions we must enumerate alternative extreme points. The feasible space of a 2-dimensional LP is shown in Figure 2 which shows the alternate solutions of an LP:

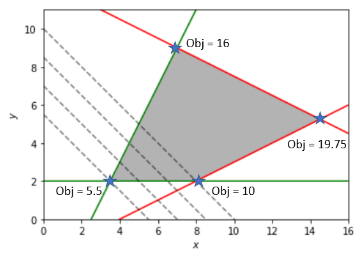


Figure 2: An example of a feasible space of a 2-dimensional LP

The stars marked in Figure 2 represent all the vertex solutions of the LP and the shaded space indicates the feasible space of the LP. As can be seen from Figure 2, the optimal solution lies on the point (3.5,2) with the objective value of 5.5. The LP shown has 3 alternate solutions with objective values of 10, 16 and 19.75. If we wish to traverse through all the vertexes and identify alternate solutions in increasing value of objective function, we need to identify the constraints that are inactive. The constraints marked in green in Figure 2 are active for the optimum value of 5.5 while the ones marked in red are inactive. To identify the next alternate solution, point (8,2) which has an objective value of 10, we identify the active and the inactive constraints of the current best solution by adding slack variables to all the inequality constraints in our LP model. If the slack variable is strictly greater than zero of a particular constraint then the constraint is inactive and a zero value of the slack variable would imply that the constraint is active. We proceed by forcing at least one inactive constraint to be active by adding cuts using binary variables introduced in our MILP formulation which allows us to traverse through all the vertex solutions.

4.2 MILP Master Problem

The general idea of the algorithm shown in Section 3.1 is to identify the inactive constraints in the LP model and to force at least one of them to become active. We implement this by adding slack variable to the inequality constraint in our model and control the activity of these constraints by adding binary variables to the model. The set of inactive constraints are identified by the value of the slack variable. Hence, if the slack variable is strictly greater than zero ($s_i > 0$ or $s_i \geq \epsilon$ where ϵ is an appropriate small number) we know that the constraint is inactive.

Let us consider the linear programming problem(2) in its general form for which the slack variables s are added shown in the following model (3)

$$\begin{array}{ll} \min & z = c^T x \\ s.t. & Ax = B \\ & Ex + Is = d \\ & s \geq 0 \\ & x \in R^n \end{array} \tag{3}$$

Let us assume we obtain the optimal solution to the LP using the simplex method or interior point method (iteration K = 0). The MILP master problem corresponding to the LP model (3) for iteration K = 1,2,3...,N is shown below. To determine the functionality of the constraints, binary variables y_i are introduced. y_i is a binary variable that indicates whether the slack variable is zero which helps us to control the activity of constraints using slack variables. The MILP model (4) is shown below in which the inequalities are written for each individual constraint:

$$\begin{array}{lll} \min & z = c^T x & \\ s.t. & Ax = B & \\ & e_i x + s_i - d_i = 0 & \forall i \in \{1, 2, ..., m\} \\ & s_i \leq M\,(1 - y_i) & \forall i \in \{1, 2, ..., m\} \\ & s_i \geq 0 & \forall i \in \{1, 2, ..., m\} \\ & \sum\limits_{i \in NB^k} y_i \geq 1 & \forall k \in \{0, ..., K - 1\} \\ & x \in R^m & \end{array} \tag{4}$$

The objective function and the equality constraints of the LP model remain unchanged in our formulation. The individual inequality constraints are reformulated as equations - $e_i x + s_i - d_i = 0$ using slack variables - s_i. We add the constraint $s_i \leq M(1 - y_i)$ which forces the slack variable of the i^{th} constraint to be ($s_i \leq 0$) when the binary variable (y_i) is set to be 1. Therefore, the constraint $s_i \leq 0$ and the non-negativity constraint on the slack variable $s_i \geq 0$ forces the slack variable to be 0 - $s_i = 0$. This forces the i^{th} constraint to be active. The set i goes from 1 to m where m is the number of inequality constraints in the LP model.

The constraint $\sum\limits_{i \in NB^k} y_i \geq 1 \quad \forall k$ is the cut that allows us to move from the current solution to the next best solution. The summation is over the indexed set NB^k. The set NB^k is the set of all slack variables that are strictly greater than 0 for iteration K. For example, if $s1 = 10$, $s2 = 12$, $s3 = 0$, and $s4 = 2$, the set NB would be - $\{1, 2, 4\}$ since these slack variables are strictly greater than 0. This set is indexed over set k which goes from 0 to $K - 1$ where K is the current iteration. Therefore, for iteration K we have added K-1 cuts to ensure that we obtain the K^{th} best solution. We start our recursive algorithm with $K = 1$ and set NB^0 as a predefined empty set. Iteration 0 is to solve the LP (3) and obtain the optimal solution.

The set NB^k is not the same as the set of Non-Basic variables for an LP solution. It is possible that some of the variables in the set of Non-Basic variables can be 0 and this leads to degenerate solutions. Therefore, we define the set NB^k in model (4) to be the set of all slack variables that are strictly greater than 0 for iteration k.

4.3 Algorithm

The recursive MILP method to find the N^{th} best solution of an LP consists of the following steps:

Step 0: The first step is to solve the LP model (3) to obtain the optimal solution. Note that we use slack variables to identify the active and inactive constraints for our algorithm.

Step 1: Set K=1, Reformulate the LP and replace all inequality constraints with equality constraints using slack variables. The inequality constraint in Model (3) is reformulated into equations by using slack variables in model (4). As in model (4) we add the constraint $s_i \leq M(1 - y_i)$ that forces the slack of the i^{th} constraint to be active ($s_i = 0$) when the binary variable (y_i) is set to be 1. The constraint $\sum\limits_{i \in NB^k} y_i \geq 1 \quad \forall k$ is the cut that allows us to move from the current solution to the next best solution.

Step 2: Record the optimal objective value, optimal solution, and determine which slack variables are non-zero

Step 3: Add a cut on these variables using the corresponding binary variables. The following constraint is used to add cuts in our recursive model:

$$\sum_{i \in NB^k} y_i \geq 1 \qquad \forall k \in \{0, ..., K-1\}$$

is defined by the set of non zero slack variables in the previous iteration. We use this set to add cuts that allow us to move from the current solution to the next best solution.

Step 4: Resolve the model (4) and record the objective value and the solution. Keep adding cuts using step 3.

Step 5: The exit block of our algorithm is when we reach the N^{th} best solution or when we get an infeasible solution in which there do not exist N solutions for the given LP. An LP problem (3) is first solved for iteration 0 to identify the current basis and optimal value. For K = 0... N-1 iterations, an MILP master problem is solved until the N^{th} best solution is reached or an infeasible solution is obtained.

The algorithm is shown in Figure 3 in the form of a block diagram:

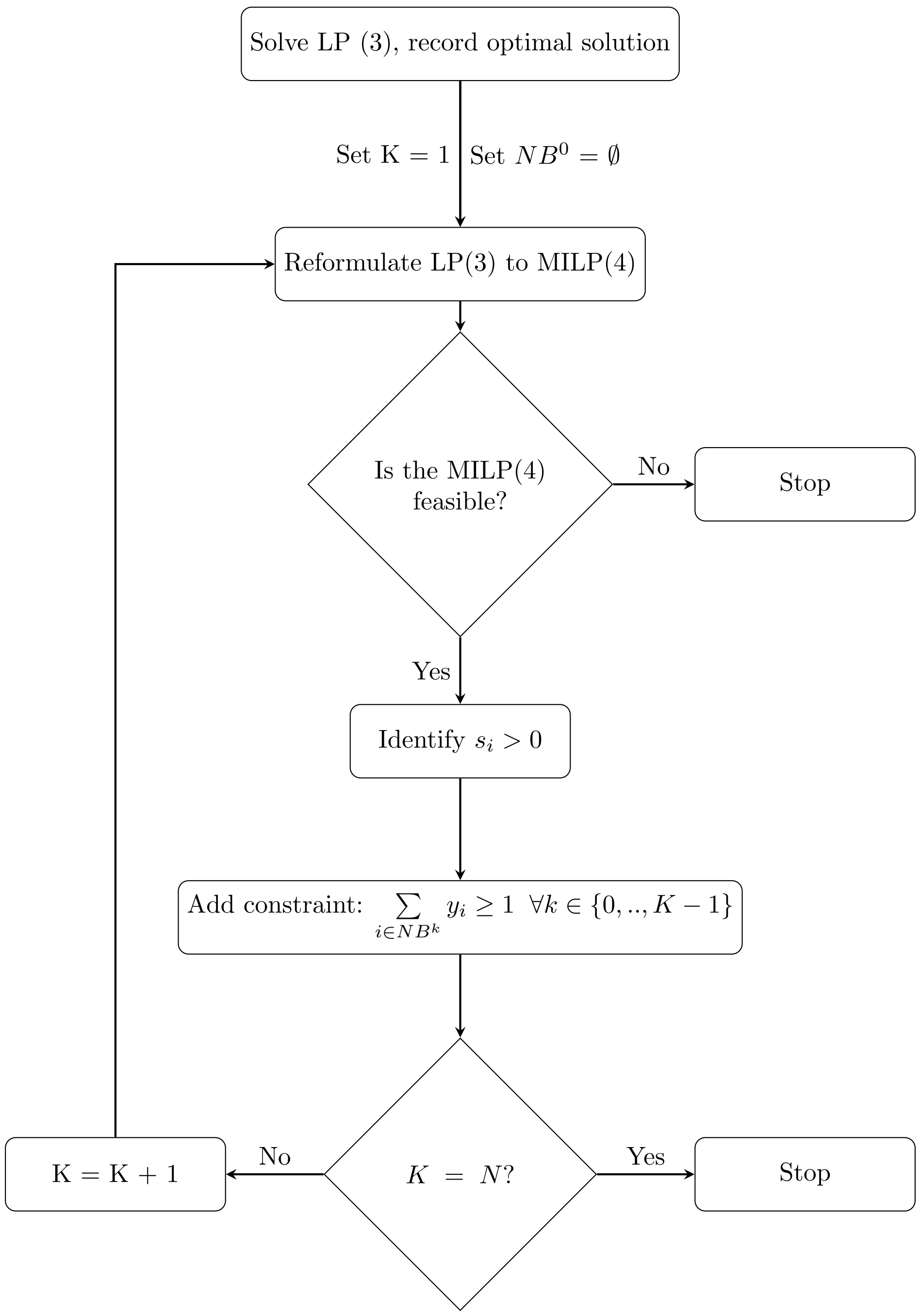

Solve LP (3), record optimal solution
Set K = 1
Set $NB^0 = \emptyset$
Reformulate LP(3) to MILP(4)
Is the MILP(4) feasible?
No
Stop
Yes
Identify $s_i > 0$
Add constraint: $\sum_{i \in NB^k} y_i \geq 1 \;\; \forall k \in \{0, .., K-1\}$
$K = N$?
No
K = K + 1
Yes
Stop

Figure 3: Algorithm for alternate solutions

5 Results

The implementation of the algorithm is illustrated through three example problems. The first example illustrates an LP problem (with alternate solutions) expanded at each iteration of the algorithm. The second example is the implementation of the illustrative supply chain problem in section 2. The third example is a large scale supply chain problem provided by Aurubis AG. The MILP model for each iteration was implemented in Pyomo (Bynum et al. 2021) in Python. For solving the MILP model, CPLEX 20.1 was used. All Pyomo and CPLEX settings were left at their default values. Appropriate computational experiments were conducted using an Intel Core i7 8565U @1.60GHz machine, with a RAM of 16GB.

5.1 Example 1: 2-dimensional LP

We illustrate the algorithm shown in Section 4 by implementing it on a 2-dimension Linear Programming problem shown in (5):

$$\begin{aligned} min \quad & z = x_1 + 2x_2 \\ s.t. \quad & x_1 \leq 8 \\ & x_2 \leq 10 \\ & x_1 + x_2 \leq 8 \\ & 3x_1 + 1.5x_2 \geq 9 \\ & x_1 - x_2 \leq 3 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned} \tag{5}$$

The corresponding feasible space of Model (5) is shown in Figure 4. The optimal solution (obj value = 3) along with the 3 alternate solutions (obj value = 10.5,12, and 16) are marked on the feasible space in Figure 4:

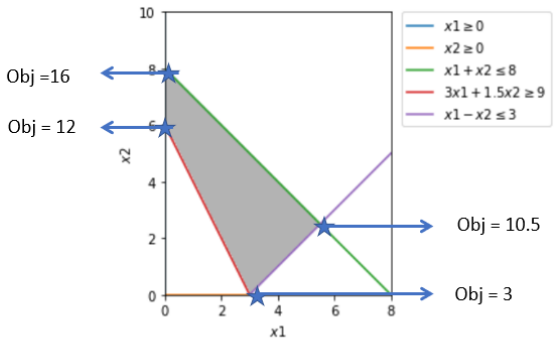


Figure 4: Feasible space of Model 4

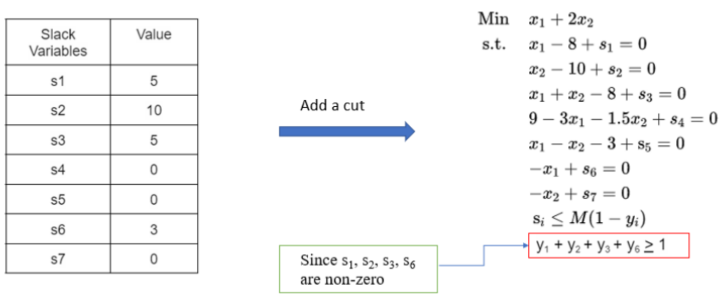
The first step of our algorithm is to Reformulate the LP to an MILP using slack variables and binary variables. We replace all inequality constraints with equations using the slack variables. We add the constraint $s_i \leq M(1 - y_i)$ to control the activity of the inequality constraints. We also add the constraint $\sum y_i \geq 1 \quad \forall i \in NB^k$ which is the cut that allows us to move from the current solution to the next best solution. The reformulated MILP is shown in Model (6):

$$\begin{aligned} Min \quad & z = x_1 + 2x_2 \\ s.t. \quad & x_1 - 8 + s_1 = 0 \\ & x_2 - 10 + s_2 = 0 \\ & x_1 + x_2 - 8 + s_3 = 0 \\ & 9 - 3x_1 - 1.5x_2 + s_4 = 0 \\ & x_1 - x_2 - 3 + s_5 = 0 \\ & -x_1 + s_6 = 0 \\ & -x_2 + s_7 = 0 \\ & s_i \leq M(1 - y_i) \qquad \forall i \in \{1, .., 7\} \\ & \sum_{i \in NB^k} y_i \geq 1 \qquad \forall k \in \{0, .., K - 1\} \end{aligned} \tag{6}$$

Iteration 0: In this iteration we solve the LP in (5) an obtain the optimal solution. We do not have any cuts in this iteration. We record the value of the slack variables for the next iteration. We get an objective value of z =

3 for $x_1 = 3$ and $x_2 = 0$ which is the optimal solution.

Iteration 1: The table shown in Figure 5 displays the slack variable values from the solution we obtain in iteration 0. In this case, s1, s2, s3, and s6 are non-zero slack variables. Therefore we obtain the set $NB^1 = \{1, 2, 3, 6\}$ and we add a cut over this set using the binary variables defined. Since we focus only on the non-zero slack variables in our algorithm, the presence of such a degenerate case does not affect our process. We obtain an objective value of 10.5

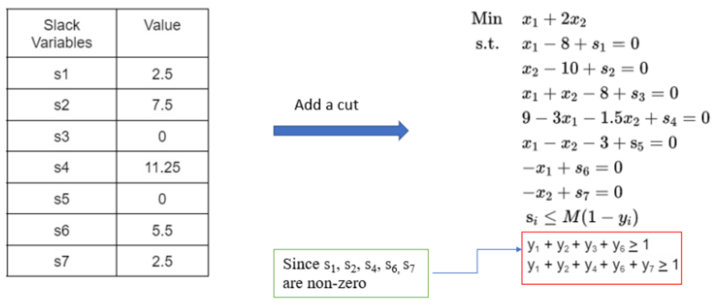


Slack Variables	Value
s1	5
s2	10
s3	5
s4	0
s5	0
s6	3
s7	0

$$\begin{aligned} \text{Min} \quad & x_1 + 2x_2 \\ \text{s.t.} \quad & x_1 - 8 + s_1 = 0 \\ & x_2 - 10 + s_2 = 0 \\ & x_1 + x_2 - 8 + s_3 = 0 \\ & 9 - 3x_1 - 1.5x_2 + s_4 = 0 \\ & x_1 - x_2 - 3 + s_5 = 0 \\ & -x_1 + s_6 = 0 \\ & -x_2 + s_7 = 0 \\ & s_i \leq M(1 - y_i) \\ & y_1 + y_2 + y_3 + y_6 \geq 1 \end{aligned}$$

Figure 5: Iteration 1: Objective Value z = 10.5

Iteration 2: The table shown in Figure 6 displays the slack variable values from the solution we obtain in iteration 1. In this case, s1, s2, s4, s6 and s7 are non-zero slack variables. Therefore we obtain the set $NB^2 = \{1, 2, 4, 6, 7\}$ and we add a cut over this set using the binary variables defined. We keep all the cuts from the previous iterations to prevent repeating solutions. We obtain an objective value of z = 12

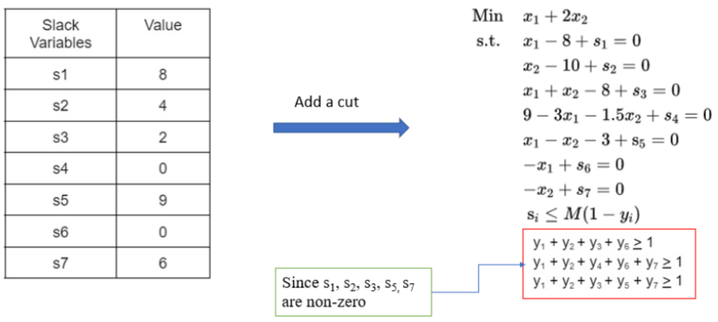


Slack Variables	Value
s1	2.5
s2	7.5
s3	0
s4	11.25
s5	0
s6	5.5
s7	2.5

$$\begin{aligned} \text{Min} \quad & x_1 + 2x_2 \\ \text{s.t.} \quad & x_1 - 8 + s_1 = 0 \\ & x_2 - 10 + s_2 = 0 \\ & x_1 + x_2 - 8 + s_3 = 0 \\ & 9 - 3x_1 - 1.5x_2 + s_4 = 0 \\ & x_1 - x_2 - 3 + s_5 = 0 \\ & -x_1 + s_6 = 0 \\ & -x_2 + s_7 = 0 \\ & s_i \leq M(1 - y_i) \\ & y_1 + y_2 + y_3 + y_6 \geq 1 \\ & y_1 + y_2 + y_4 + y_6 + y_7 \geq 1 \end{aligned}$$

Figure 6: Iteration 2: Objective Value z = 12

Iteration 3: The table shown in Figure 7 displays the slack variable values from the solution we obtain in iteration 2. In this case, s1, s2, s3, s5 and s7 are non-zero slack variables. Therefore we obtain the set $NB^3 = \{1, 2, 3, 5, 7\}$ and we add a cut over this set using the binary variables defined. We keep all the cuts from the previous iterations to prevent repeating solutions. This is the final iteration and we obtain an objective value of z = 16. Note that if we continue to add a cut on the positive slack variables we will get an infeasible solution as there are no more vertex solutions remaining. Hence, we terminate the algorithm after the 3^{rd} iteration.



Slack Variables	Value
s1	8
s2	4
s3	2
s4	0
s5	9
s6	0
s7	6

$$\begin{aligned} \text{Min} \quad & x_1 + 2x_2 \\ \text{s.t.} \quad & x_1 - 8 + s_1 = 0 \\ & x_2 - 10 + s_2 = 0 \\ & x_1 + x_2 - 8 + s_3 = 0 \\ & 9 - 3x_1 - 1.5x_2 + s_4 = 0 \\ & x_1 - x_2 - 3 + s_5 = 0 \\ & -x_1 + s_6 = 0 \\ & -x_2 + s_7 = 0 \\ & s_i \leq M(1 - y_i) \end{aligned}$$

$$\begin{aligned} & y_1 + y_2 + y_3 + y_6 \geq 1 \\ & y_1 + y_2 + y_4 + y_6 + y_7 \geq 1 \\ & y_1 + y_2 + y_3 + y_5 + y_7 \geq 1 \end{aligned}$$

Since s_1, s_2, s_3, s_5, s_7 are non-zero

Figure 7: Iteration 3: Objective Value = 16

We obtain the following solution pool for the 2-d LPP problem:

Iteration	x_1	x_2	*Obj*	*Model*
0	3	0	3	LP
1	5.5	2.5	10.5	MILP with cuts
2	0	6	12	MILP with cuts
3	0	8	16	MILP with cuts

Table 1: Solution pool for Example 1

We have illustrated an improved recursive MILP approach that will not only work for obtaining alternate optimal solutions but it will produce a solution pool of all the alternate solutions in the increasing order of value of the objective function.

5.2 Example 2: Supply Chain Problem

Let us consider the supply chain model shown in Figure 8. It is representative of a small supply chain network with 3 plants and 2 customers to demonstrate the application and effectiveness of our algorithm.

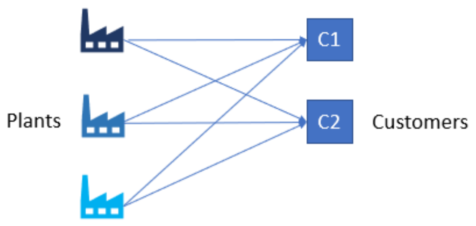


Figure 8: supply chain problem

The optimization is subjected to the following constraints: Equation (7b) states the capacity constraint of each plant. Equation (7c) is a demand satisfaction constraint that ensures the sum flow of products from all plants is greater than or equal to the demand of the customer. Equation (7d) are the variable bounds. The objective the model is to minimize costs (7a).

$$\begin{aligned} \min \quad & Z = \sum_i \sum_j C_i * F_{ij} && (7a) \\ s.t \quad & \sum_j F_{i,j} \leq Cap_i \quad \forall i && (7b) \\ & \sum_i F_{i,j} = D_j \quad \forall j && (7c) \\ & \sum_j F_{i,j} \geq 0 \quad \forall i, \forall j && (7d) \end{aligned}$$

We have 3 sets of parameters defined. Tables 2, 3, and 4 highlight the parameters. Table 2 has the transportation costs associated between the suppliers and the customers. Table 3 and 4 has the plant capacities and the customer demands respectively.

	C_1	C_2
P_1	6000	3000
P_2	4000	5000
P_3	5000	4000

Table 2: Transportation cost ($ per unit) between suppliers and customers.

	Capacity
P_1	60
P_2	50
P_3	40

Table 3: Plant capacities.

	Demand
C_1	50
C_2	60

Table 4: Customer Demand.

The LP supply chain model is optimized by maximizing profit subject to the constraints shown in Model (7). When the optimization is run with the specified parameters, we get the following cost $ 380K. The optimal solution for the optimization is given on the boxes in the arrows going from one node to another in iteration 1. The algorithm is applied and the cuts have been added at every iteration to move to the next vertex solution. The first 4 iterations are shown in Figure 9. The optimization is run until all the vertex solutions are exhausted. The solution pool obtained is shown in Table 5 and the objective value at every iteration is visualized in Figure 10. Note that solutions at iterations (1,2), (4,5,6), (7,8), and (9,10) have the same value of the objective function and hence rae degenerate solutions.

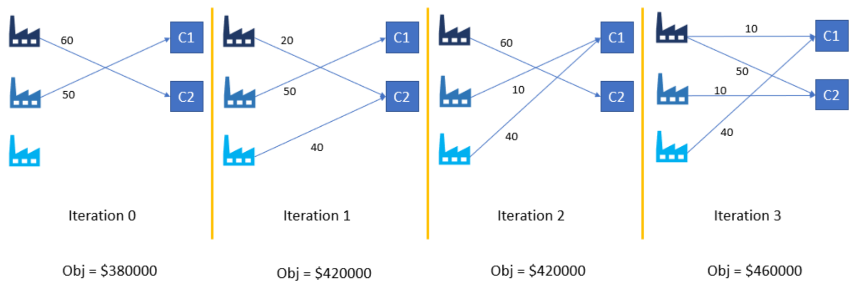


Figure 9: Solutions for the first 4 iterations

Iteration	$F_{1,1}$	$F_{1,2}$	$F_{2,1}$	$F_{2,2}$	$F_{3,1}$	$F_{3,2}$	*ObjectiveValue*($)
0	0	60	50	0	0	0	**380K**
1	0	20	50	0	0	40	**420K**
2	0	60	10	0	40	0	**420K**
3	10	50	0	10	40	0	**460K**
4	20	0	30	20	0	40	**500K**
5	0	20	10	40	40	0	**500K**
6	40	20	10	0	0	40	**500K**
7	10	10	0	50	40	0	**540K**
8	50	10	0	10	0	40	**540K**
9	50	0	0	20	0	40	**560K**
10	20	0	0	50	30	10	**560K**
11	50	10	0	50	0	0	**580K**
12	50	0	0	50	0	10	**590K**
13	–	–	–	–	–	–	**Inf**

Table 5: Solution pool for example 2.

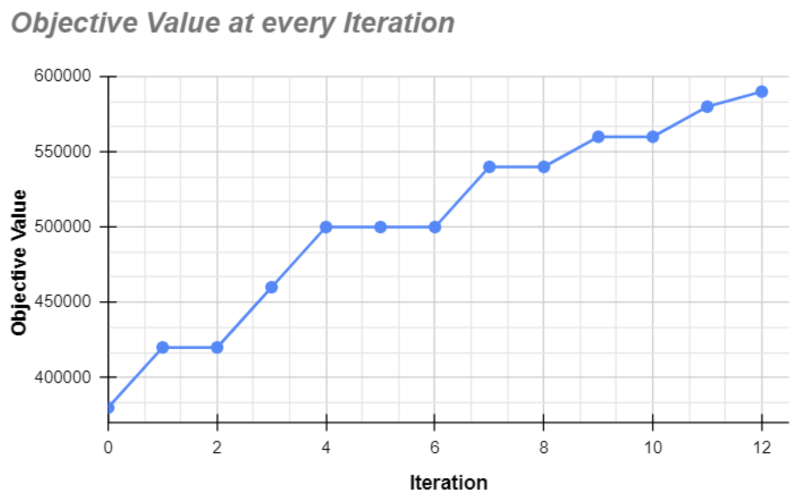


Figure 10: Objective value at every iteration

The objective value at every iteration is given by Figure 10. We have successfully demonstrated the algorithm and obtained 13 alternate vertex solutions for the supply chain problem. This gives the user the ability to choose among these solutions typically considering the practical considerations that are not explicitly included in the model. Hence, we have solved the problem to develop a procedure or algorithm that can provide the user with a solution pool of N best solutions of an LP model. In the next subsection, we implement on a large scale supply chain model.

5.3 Example 3: Large Scale Supply Chain Problem

5.3.1 Problem Set-up

Let us consider the supply chain model shown in Figure 11. It is representative of a supply chain model for Aurubis having 13 plants. The brown arrows indicates plants that can receive raw materials. The yellow arrow indicates the plants that produce products. The blue arrows indicates the intermediate flows between different plants. Note that some plants have multiple yellow arrows. Each yellow arrow denotes a unique product produced at that plant. Each raw material, intermediate flow and product is composed of different elements. The sets rm, p, elem, and prod represent the raw materials, plants, elements, and products. The set of links is a set of tuples to represent the intermediate flows between plants. For example, the tuple (1,2) represents the intermediate flow from plant 1 to plant 2. There are 3 variables in our model to define the different kinds of flows - f_1, f_2, and f_3. The flow f_1 is the flow of raw materials to plants. The flow f_2 is the intermediate flow for each link. The flow f_3 is the flow of products from plants.

The parameters defined for the problem are shown in Table 6

Parameter	Definition
Avail(rm)	The availability of each raw material (tons)
analysis(rm,elem)	The composition of each element in the raw material
rev(rm)	The revenue obtained by selling 1 ton of each raw material
rev(prod)	The revenue obtained by selling 1 ton of each product
ded(rm,elem)	The amount of discount we get per element in the raw material
loss(prod,elem)	The amount of discount we give per element in the product
price(elem)	The price of each element
Min_spec(prod,elem)	The minimum amount of each element in the product
Max_spec(prod,elem)	The maximum amount of each element in the product
Cap(plant)	The capacity of each plant
D(prod)	The demand of each product

Table 6: Parameters defined for the supply chain problem

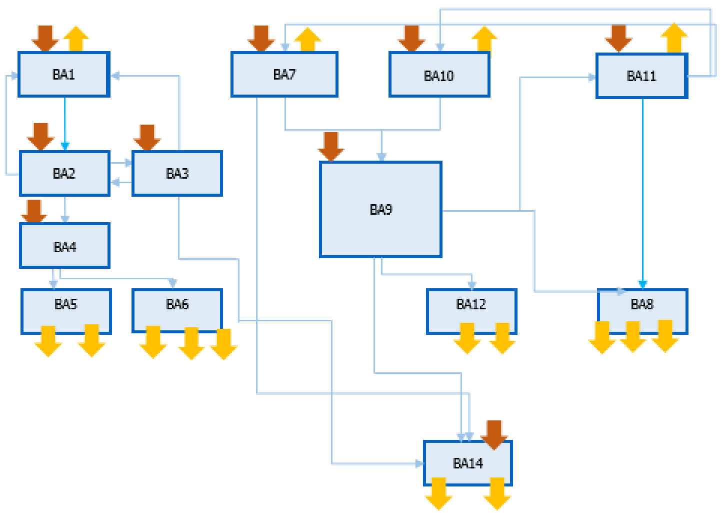


Figure 11: Large scale supply chain example problem

Figure 11 shows the supply chain problem with 13 plants. There are 167 raw materials and 16 unique products shown by the brown and yellow arrows respectively. The blue arrows indicate the intermediate flows between plants. Each of these flow is comprised of 25 different elements.
The optimization is subjected to the following constraints: Equation (8b) states that any amount of an element entering a plant is equal to the amount of that element leaving the plant. Equation (8c) is a demand satisfaction constraint that ensures the sum flow of products from all plants is greater than or equal to the demand of that product. Equation (8d) states that the total amount of raw materials and intermediate flows entering a plant is less than the capacity of the plant. Equation (8e) is the availability constraint of the raw material. Equation (8f) and (8g) represents the product specification constraint of each product. Equation (8h), (8i), and (8j) are the variable bounds. The objective the model is to maximise profit (8a). The first 3 terms of the objective are the revenue from the product, the raw material deductions and the revenue from the raw materials while the 4th term is the losses associated with each product.

$$\max \quad Z = \sum_{prod}\sum_{elem} f_3(prod, elem) * rev(prod) + \sum_{elem}\sum_{p}\sum_{rm} f_1(rm,p) * analysis(rm, elem) * ded(rm, elem) * price(elem)$$

$$\sum_{p}\sum_{rm} f_1(rm,p) * rev(rm) - \sum_{elem}\sum_{prod} f_3(prod, elem) * loss(prod, elem) * price(elem) \tag{8a}$$

$$s.t. \quad \sum_{rm} f_1(rm,p) * analysis(rm, elem) + \sum_{p'} f_2\left(p', p, elem\right) = \sum_{p'} f_2\left(p, p', elem\right) + \sum_{prod} f_3(prod, elem) \quad \forall p, \forall elem \tag{8b}$$

$$\sum_{elem} f_3(prod, elem) \geq Demand(prod) \quad \forall prod \tag{8c}$$

$$\sum_{rm} f_1(rm,p) + \sum_{p', elem} f_2\left(p', p, elem\right) \leq Capacity(p) \quad \forall p \tag{8d}$$

$$\sum_{p} f_1(rm,p) \leq Avail(rm) \quad \forall rm \tag{8e}$$

$$f_3(prod, p) \geq MinSpec(prod, elem) * \sum_{elem} f_3(prod, elem) \quad \forall prod, \forall elem \tag{8f}$$

$$f_3(prod, p) \leq MaxSpec(prod, elem) * \sum_{elem} f_3(prod, elem) \quad \forall prod, \forall elem \tag{8g}$$

$$f_1(rm,p) \geq 0 \quad \forall rm, \forall p \tag{8h}$$

$$f_2(link, elem) \geq 0 \quad \forall link, \forall elem \tag{8i}$$

$$f_3(prod, elem) \geq 0 \quad \forall prod, \forall elem \tag{8j}$$

5.3.2 Optimal Solution of the LP

The LP supply chain model is optimized by maximizing profit subject to the constraints shown in Model (8). When the optimization is run with the specified parameters, we get the following cost €266.898 million. The solution for the optimization is given on the boxes in the arrows going from one node to another. The brown arrows indicates plants that can receive raw materials. The yellow arrow indicates the plants that produce products. The blue arrows indicates the intermediate flows between different plants. Note that some plants have multiple yellow arrows. Each yellow arrow denotes a unique product produced at that plant. Each raw material, intermediate flow and product is composed of different elements. Only the flow of raw materials to the plants (f_1) and the flow of products (f_3) are shown for simplicity. The flows is in tonnes.

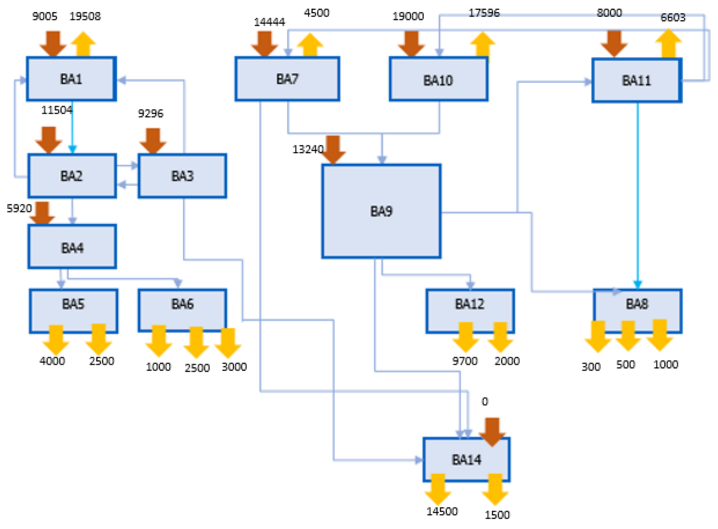


Figure 12: Optimal solution of the supply chain LP

The objective value of the optimal solutions is €266.898 million. The model statistics is given by Table 7.

Continuous Variables	523
Constraints	3047
Solver	Cplex
Solve Time	0.11s
Objective Value	€266.898 million

Table 7: Model Statistics for the LP supply chain problem

5.3.3 Generating alternate solutions

As described in Section 4.3 of this paper, we first reformulate the LP and ad slack variables to all the inequality constraints of our model. The reformulated model is given by model (9). Note that the objective function (9a) and the constraint (9b) remain unchanged. We also add the traditional Big-M constraints to the slack variables shown in (9k) - (9r).

$$\max \quad z = \sum_{prod}\sum_{elem} f_3(prod, elem) * rev(prod) + \sum_{elem}\sum_{p}\sum_{rm} f_1(rm, p) * analysis(rm, elem) * ded(rm, elem) * price(elem)$$

$$\sum_{p}\sum_{rm} f_1(rm, p) * rev(rm) - \sum_{elem}\sum_{prod} f_3(prod, elem) * loss(prod, elem) * price(elem) \tag{9a}$$

$$s.t. \quad \sum_{rm} f_1(rm, p) * analysis(rm, elem) + \sum_{p'} f_2\,(p', p, elem) = \sum_{p'} f_2\,(p, p', elem) + \sum_{prod} f_3(prod, elem) \quad \forall p, \forall elem \tag{9b}$$

$$Demand(prod) - \sum_{elem} f_3(prod, elem) + s_1(prod) = 0 \quad \forall prod \tag{9c}$$

$$\sum_{rm} f_1(rm, p) + \sum_{p', elem} f_2\,(p', p, elem) - Capacity(p) + s_2(p) = 0 \quad \forall p \tag{9d}$$

$$\sum_{p} f_1(rm, p) - Avail(rm) + s_3(rm) = 0 \quad \forall rm \tag{9e}$$

$$MinSpec(prod, elem) * \sum_{elem} f_3(prod, elem) - f_3(prod, p) + s_4(prod, elem) = 0 \quad \forall prod, \forall elem \tag{9f}$$

$$f_3(prod, p) - MaxSpec(prod, elem) * \sum_{elem} f_3(prod, elem) + s_5(prod, elem) = 0 \quad \forall prod, \forall elem \tag{9g}$$

$$s_6(rm, p) - f_1(rm, p) = 0 \quad \forall rm, \forall p \tag{9h}$$

$$s_7(link, elem) - f_2(link, elem) = 0 \quad \forall link, \forall elem \tag{9i}$$

$$s_8(prod, elem) - f_3(prod, elem) = 0 \quad \forall prod, \forall elem \tag{9j}$$

$$s_1(prod) \leq M * (1 - y_1(prod)) \quad \forall prod \tag{9k}$$

$$s_2(p) \leq M * (1 - y_2(p)) \quad \forall p \tag{9l}$$

$$s_3(rm) \leq M * (1 - y_3(rm)) \quad \forall rm \tag{9m}$$

$$s_4(prod, elem) \leq M * (1 - y_4(prod, elem)) \quad \forall prod, \forall elem \tag{9n}$$

$$s_5(prod, elem) \leq M * (1 - y_5(prod, elem)) \quad \forall prod, \forall elem \tag{9o}$$

$$s_6(rm, p) \leq M * (1 - y_6(rm, p)) \quad \forall rm, \forall p \tag{9p}$$

$$s_7(link, elem) \leq M * (1 - y_7(link, elem)) \quad \forall link, \forall elem \tag{9q}$$

$$s_8(prod, elem) \leq M * (1 - y_8(prod, elem)) \quad \forall prod, \forall elem \tag{9r}$$

We follow the algorithm shown in section 4.3 and solve the model for the optimal solution. Next, we identify the slack variables that are strictly greater than zero and add a cut over this set of slack variables. Generating alternate solutions for a large scale supply chain problems like these can lead to solutions that are very close to the optimal solution. Therefore, we add additional cuts only on the decision variables - f_1 & f_2 to ensure we get significantly different solutions for practical purposes.

We solved the model for 10 iterations to identify 10 alternate solutions for our supply chain problem. The solution obtained at the 10^{th} iteration is given by Figure 13. The objective value z = €266.686 million

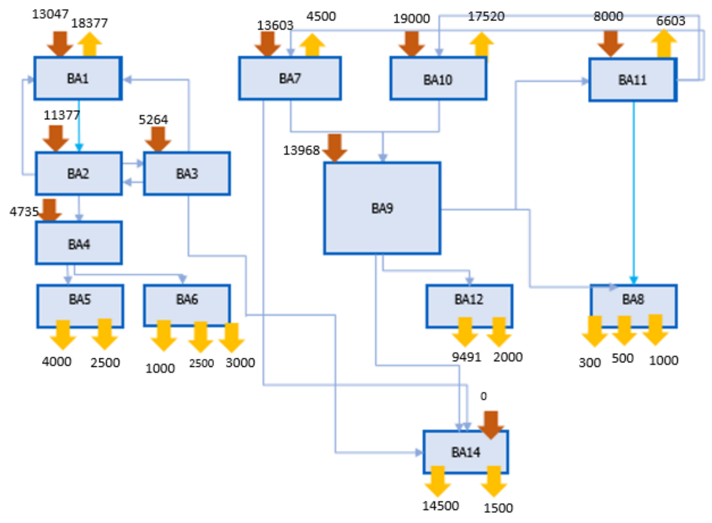


Figure 13: 10th Alternate solution of the supply chain problem

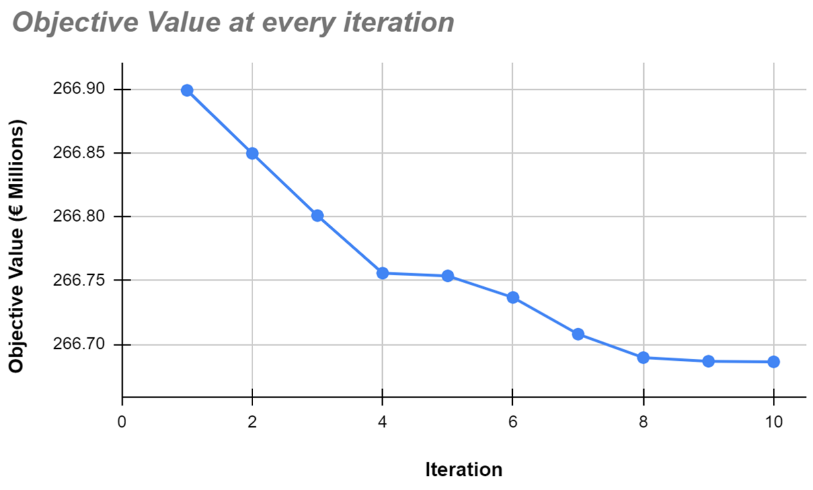


Figure 14: Objective Value of all 10 iterations

The objective value at every iteration is given by Figure 14. We have successfully demonstrated the algorithm and obtained 10 alternate solutions for the supply chain problem. We also note that all 10 alternate solutions are within 0.1% of the optimal solution. This gives the user the ability to choose among these solutions typically considering practical considerations that are not explicitly included in the model. Hence, we have solved the problem to develop a procedure or algorithm that can provide the user with a solution pool of N best solutions of an LP instead of just reporting a single optimal solution.

We solved the model using 3 different solvers to compare the performance of the algorithm, Cplex, Gurobi, and Glpk (Makhorin, A., 2008) for our solutions. The total solution time and the model statistics is given by Table 8.

Continuous Variables	2335
Discrete Variables	996
Constraints	5055
Solve Time using Gurobi	46.67s
Solve Time using Cplex	53.96s
Solve Time using Glpk	54.13s
Objective Value of the 1st *iteration*	€266.898 million
Objective Value of the 10th *iteration*	€266.686 million

Table 8: Model statistics for the recursive MILP formulation

6 Conclusion

In this paper, we have addressed the problem to find alternate solutions of an LP. Our goal is to find the optimal solution as well as alternate solutions with the same objective value or with increasing objective value. To resolve this problem, we provide a recursive MILP algorithm that generates alternate solutions with the same or increasing value of the objective function by identifying inactive constraints at the current solution and forces atleast one to be active using binary variables.

We demonstrate the application and effectiveness of our proposed algorithm in section 5. We illustrate the algorithm on a 2-dimensional LP in section 5.1. Next, we show a small scale supply chain problem and generate alternate solutions for the problem. We also demonstrate its effectiveness on a large scale supply chain problem in section 5.3 provided by Aurubis AG. We generate 10 different alternate solutions within 0.1% of the optimal solution and generates a solution pool instead of a single optimal solution. This allows the user to choose among these solutions given the practical considerations that are not explicit in the model.

References

Bynum, Michael L., Gabriel A. Hackebeil, William E. Hart, Carl D. Laird, Bethany L. Nicholson, John D. Siirola, Jean-Paul Watson, and David L. Woodruff (2021). Pyomo - Optimization Modeling in Python. Third Edition Vol. 67. Springer, 2021.

Cafaro, D.C. and I.E. Grossmann (2014), “Strategic Planning, Design and Development of the Shale Gas Supply Chain Network,” AIChE J. 60, 2122-2142 .

Charnes, A., W.W. Cooper (1957), ”Management Models and Industrial Applications of Linear Programs,” Management Science, 4, 38-91

Dantzig, George B. (1982), ”Reminiscences about the origins of linear programming,” Operations Research Letters., 1, 43–48.

Gass, S.I. (2003), ”Linear Programming Methods and Applications,” 5th Ed, Dover Publications.

Grossmann, I.E. (2005) , ”Enterprise-wide optimization: A new frontier in process systems engineering,”AIChE Journal, 51, 1846-1857.

Harjunkoski, I., Maravelias, C.T., Bongers, P., Castro, P., Engell, S., Grossmann, I.E., Hooker, J., Mendez, C., Sand, G. and Wassick, J., (2014), “Scope for Industrial Applications of Production Scheduling Models and Solution Methods,” Computers and Chemical Engineering, 62, 161-193 .

Karmarkar, N.K. (1984), ”A new polynomial-time algorithm for linear programming,” Combinatorica, 4. 373–398.

Lee, S., C. Phalakornkule, M.D. Domach and I.E. Grossmann (2000), ”Recursive MILP Model for finding all the Alternate Optima in LP models for Metabolic Networks,” Computers and Chemical Engineering 24, 711-716).

Luenberger, D. G., & Ye, Y. (2008). Linear and nonlinear programming. Springer Science & Business Media.

Makhorin, A.: GLPK—GNU linear programming kit.http://www.gnu.org/software/glpk/glpk.html (2008)

Maravelias, C. T., &; Sung, C. (2009), ”Integration of production planning and scheduling: Overview, challenges and opportunities,” Computers and Chemical Engineering, 51, 1919-1930.

Matheiss, T. H., & Rubin, D. S. (1980). A survey and comparison of methods for finding all vertices of convex polyhedral sets. Mathematics of Operations Research, 5(2), 167-185

Swart, G. (1985). ”Finding the convex hull facet by facet”, Journal of Algorithms, 6, 17-48.

Yang, T., Geng, N., Li, S., & Ma, L. (2015). Application of linear programming to vehicle routing problem. Journal of Traffic and Transportation Engineering (English Edition), 2(6), 353-361.

You, F. and I.E. Grossmann (2008), “Design of Responsive Process Supply Chains under Demand Uncertainty,” Computers & Chemical Engineering, 32, 3090-3111 .