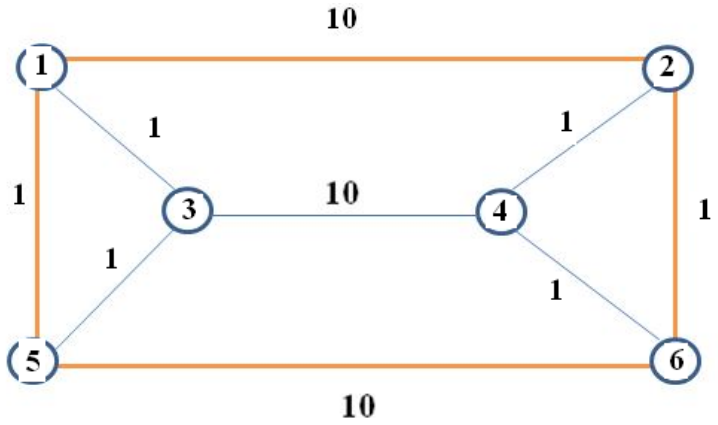
1.5 The Travelling Salesman Problem

The most common CO problems are those that organize a collection of locations (or jobs, customers, points and so on) into sequences or routes, which are widely known as Routing problems (مسائل التوجيه). Travelling Salesman Problem (TSP) is the simplest and most famous one, which seeks *a minimum-total-length route visiting every point in a given set exactly once*.

Example



The figure (to be given in lecture) shows 6 nodes and all the available links among them, with numbers on edges indicating transit times. We wish to find the shortest route visiting all nodes exactly once using only links shown on the graph. Formulate an IP for this problem.

SOLUTION

Step 1: Definition of the Decision Variables

$$x_{ij} = \begin{cases} 1, & if\ route\ (i,j) \in A\ is\ taken \\ 0, & otherwise \end{cases} \quad , \forall\ (i,j) \in A, \quad \forall\ i < j$$

We define x_{ij} only for i < j, as it avoids the duplication of arcs *(i, j) and (j, i)*. Let *A* be a set of arcs.

Step 2: Define the Constraints

As no node is to be visited more than once, we will enter and left it exactly once.

$$x_{12} + x_{13} + x_{15} = 2 \qquad (node\ 1)$$
$$x_{12} + x_{24} + x_{26} = 2 \qquad (node\ 2)$$
$$x_{13} + x_{34} + x_{35} = 2 \qquad (node\ 3)$$

$$x_{24} + x_{34} + x_{46} = 2 \qquad (node\ 4)$$
$$x_{15} + x_{35} + x_{56} = 2 \qquad (node\ 5)$$
$$x_{26} + x_{46} + x_{56} = 2 \qquad (node\ 6)$$

Step 3: Define the Objective Function
We want to minimize the total tour length:

$$\min\ 10x_{12} + x_{13} + x_{15} + x_{24} + x_{26} + 10x_{34} + x_{35} + x_{46} + 10x_{56}$$

However, this model does NOT capture the problem *entirely*. To see this, let us have a look at the optimal solution of the above model:



Tours shown in this figure satisfies all constraints listed above, but it has two disconnected subtours other than a single route through all nodes. Therefore, we need a constraint (or a set of constraints) to remove subtours. One form of subtour elimination constraints is obtained from connecting all non-trivial partitions of nodes, e.g. {1,3,5}, {2,4,6}. (Non-trivial means that the set has more than a single node in it.)

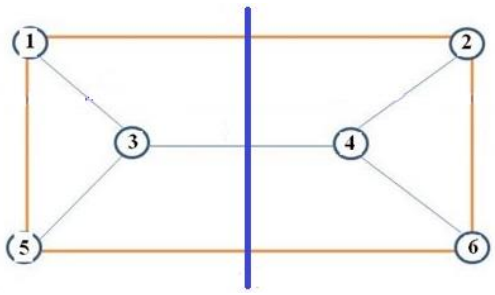
travelling salesman problem suggests deleting subtours to get the right solution.

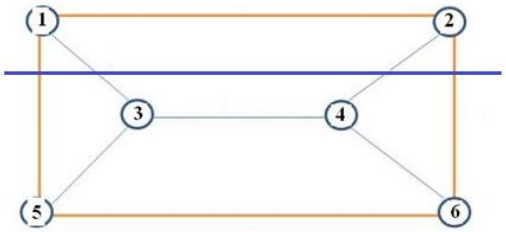
Let S represents a non-trivial subset of nodes which can possibly form a cycle. We need

> *Every tour must cross between points in* S *and points outside at least twice.*

For example, to remove the subtours mentioned above, we need:

$$x_{12} + x_{34} + x_{56} \geq 2 \qquad \text{, (partition } \{1,3,5\}, \{2,4,6\}\text{)}$$





In general, we need to include this restriction for every possible partition:
, (partition {1,2}, {3,4,5,6})
, (partition {3,4}, {1,2,5,6})
, (partition {1,2,3,4}, {5,6})

Note that {1,3}, {2,4,5,6} is not a valid partition as nodes 2,4,5,6 cannot form a cycle with existing arcs. The same for other combinations.

Therefore, a general IP model for the **TSP** is

$$\boldsymbol{min} \quad \sum_{\boldsymbol{i}} \sum_{j\backslash(i,j)\in A} c_{ij}x_{ij}$$

$$\boldsymbol{s.t.} \quad \sum_{k\backslash(k,i)\in A} x_{ki} + \sum_{j\backslash(i,j)\in A} x_{ij} = \mathbf{2}, \qquad \forall i \in N$$

$$\sum_{i\in S} \sum_{k\notin S\backslash(k,i)\in A} x_{ki} + \sum_{i\in S} \sum_{j\notin S\backslash(i,j)\in A} x_{ij} \geq \mathbf{2}, \qquad \forall S,$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A,$$

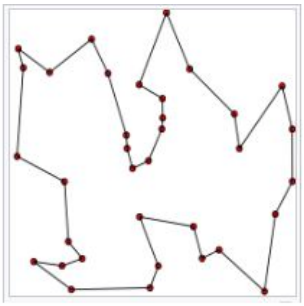
where
• c_{ij} is the direct travel time from city i to city j.
• N is the set of nodes;
• A is the set of edges ((i, j), $i < j$);
• S is a valid partition of nodes.

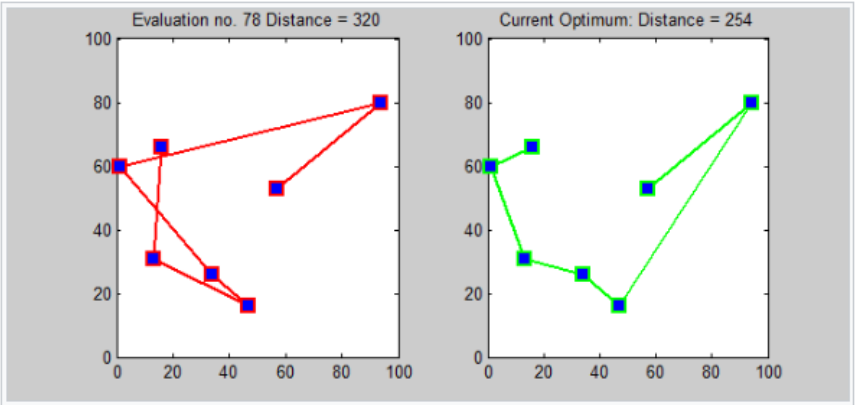
• Exactly one city *k* must be visited immediately before city *i*. Exactly one city *j* must be visited immediately after city *i*.

.

Remarks

• There are at most $2^{|N|}$ constraints in the above model to remove all subtours, where $|N|$ states for the cardinality of node set *N*.
• A requisite model for the TSP can be very large.
• There are many different formulations for the TSP and none of them are straightforward. What we have introduced is the (conceptually) simplest one.
• There exits formulations that require fewer constraints to ensure no subtours, but the constraints number is still very high for realistic problems.
• There is still no satisfactory solution to large TSPs.
• Variations include Asymmetric TSP, Multiple-routes TSP.





Solution of a TSP with 7 cities using a simple Branch and bound algorithm. Note: The number of permutations is much less than Brute force search

1.6 Binary Variables and Conjunctive Normal Form

Binary variables are $0-1$ variables which are commonly used in modelling logical conditions, for example

• In **Facility Location model** we have seen conditions like:
If no facility is placed at site j, *it will not be possible to satisfy customer* i*'s demand from site j*, which is represented by constraint:

$$y_{ij} \leq x_j, \qquad \forall i \in I, \qquad j \in N;$$

• In **Set-covering model** that:
At least there is ONE set containing member i, which is corresponding to constraint: (either this or that or both)

$$\sum_{j=1}^{m} a_{ij}\, x_j \geq 1, \qquad \forall i \in N.$$

In Boolean algebra notation the set of connectives includes:

• Inclusive OR denoted by ∨ (Ex: x>0 or y=0)
• Conjunctive AND denoted by ∧
• Negation NOT denoted by ¬ (or ~)
• Implication (IF-THEN) denoted by ⟹
(Ex: if x > 0 then y = 0 means $x > 0 \Longrightarrow y = 0$)
• Equivalence (IF-AND-ONLY-IF) denoted by ⟺
These connectives are used to connect propositions *P*, *Q*, *R*,
e.g. $x > 0, y = 0, z \leq 1$ respectively.

For example, let P stand for *I eat a whole cheesecake* and Q for *I will be full*. The proposition

• *IF I eat a whole cheesecake THEN I will be full* can be represented as $P \Rightarrow Q$;

• *I do NOT eat a whole cheesecake* corresponds to $\neg P$.
Note that $P \Rightarrow Q$ does NOT implies $\neg P \Rightarrow \neg Q$.

Now consider two events E1 and E2. Define **indicator variables**

$$x_i = \begin{cases} 1, & if \;\; E_i \;\; ocurrs \\ 0, & otherwise \end{cases} \qquad ,i = 1, 2.$$

Then we have
• $E1 \vee E2$: $x_1 + x_2 \geq 1$;
• $E1 \wedge E2$: $x_1 + x_2 = 2$ or $x_1 = x_2 = 1$ or $x_1 x_2 = 1$;
• ¬E1: $x_1 = 0$ *or* $1 - x_1 = 1$;
• $E1 \Longrightarrow E2$: $x_1 \leq x_2$ (means $\begin{cases} x_1 = 1, \; x_2 = 1 \\ x_1 = 0, \; x_2 = 0 \; or \; 1 \end{cases}$)
• $E1 \Leftrightarrow E2$: $x_1 = x_2$
Note that $E1 \Longrightarrow E2$ is actually equivalent to $(\neg E1) \vee E2$, as the latter implies:

$$1 - x_1 + x_2 \geq 1$$

Exercise
Write down the mathematical formulation with binary variables representing the following propositions:

1. *Event 2 can occur only if event 1 occurs*;
2. *None of event 1 nor event 2 can occur*;
3. *Either event 1 or event 2 occurs but not simultaneously*;
4. *Event 2 can occur only if event 1 does not occur*.

Definition 1.2 A propositional logic expression with all connectives involved being ∨ ($Q_1 \vee Q_2 \vee \ldots \vee Q_n$) with the Q is being clauses involving only ∧ connectives, is said to be in the **disjunctive normal form.**
When all connectives are ∧ ($Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n$)with the $Q_i s$ being clauses involving only ∨ connectives, it is said to be in the **conjunctive normal form.**

Conjunctive normal form directly gives rise to a set of (simultaneous) constraints.

Procedure: Conjunctive Normal Form

Step 1: Replace the implication by its equivalent disjunction:

$Q_1 \Rightarrow Q_2$ becomes $(\neg Q_1) \vee Q_2$

Step 2: Apply De Morgan's Law to bracketed expression:

$\neg(Q_1 \wedge Q_2)$ becomes $(\neg Q_1) \vee (\neg Q_2)$

$\neg(Q_1 \vee Q_2)$ becomes $(\neg Q_1) \wedge (\neg Q_2)$

Step 3: Distribute the OR operator over the AND:

$(Q_1 \wedge Q_2) \vee Q_3$ becomes $(Q_1 \vee Q_3) \wedge (Q_2 \vee Q_3)$

Example
Write the logic expression (P1 ∧ P2) ∨ P3) ⇒ P4 ∨ P5 into conjunctive normal form.

SOLUTION

Step 1: Replace implication with an equivalent disjunction

$$\neg\,[(P1 \wedge P2) \vee P3\,] \vee (P4 \vee P5)$$

Step 2: Apply De Morgan's Law

$$[\neg\,(P1 \wedge P2) \wedge (\neg\,P3)\,] \vee (P4 \vee P5)$$

$$[\,((\neg P1) \vee (\neg\ P2)) \wedge (\neg\,P3)\,] \vee (P4 \vee P5)$$

Step 3: Distribute the OR operator over the AND from left to right

$$[\,((\neg P1) \vee (\neg\ P2)) \vee (P4 \vee P5)] \wedge [(\neg\,P3)\ \vee(\,P4 \vee P5)]$$

Now all connectives are the same in each square bracket, remove them we have:

$$\underset{Q1}{[(\neg P1) \vee (\neg P2) \vee P4 \vee P5]} \wedge \underset{Q2}{[(\neg P3) \vee P4 \vee P5]}$$

which is in the **conjunctive normal form** $Q1 \wedge Q2$, and

$$Q1 = (\neg P1) \vee\ (\neg P2) \vee P4 \vee P5$$

$$Q2 = (\neg P3) \vee P4 \vee P5$$

By introducing binary variables

$$x_i = \begin{cases} 1, & if\ P_i\ occurs \\ 0, & otherwise \end{cases}, \quad i = 1, \ldots, 5 \text{ above}$$

composite clauses Q1 and Q2 can be written as linear constraints of the form:

$$Q1 : (1 - x_{1)} + (1 - x_2) + x_4 + x_5 \geq 1$$

$$Q2 : (1 - x_3) + x_4 + x_5 \geq 1$$

Hence, the original implication can be equivalently written as constraints:

$$x_1 + x_2 - x_4 - x_5 \leq 1$$

$$x_3 - x_4 - x_5 \leq 0$$

Exercise

Write the proposition using **Binary variables**:

• *If either of products A or B (or both) are manufactured, then at least one of products C, D or E must also be manufactured.*

• *We can select to manufacture at least one of products A, B and C. But if C is manufactured then we must also manufacture product D.*

Indicator Variables: Hint

We have already seen this trick here and there, but let's look at the details. An *indicator variable* is a binary variable that indicates whether or not a constraint is satisfied. Given a constraint $\sum_{j\in N} a_j x_j \leq\geq b$, let M and m denote respectively an UB and a LB on the value of $\sum_{j\in N} a_j x_j - b$, and ϵ the tolerance.

- $(\delta = 1 \implies \sum_{j\in N} a_j x_j \leq b) \iff \sum_{j\in N} a_j x_j \leq b + M(1-\delta)$
- $(\sum_{j\in N} a_j x_j \leq b \implies \delta = 1) \iff \sum_{j\in N} a_j x_j \geq b + \epsilon + \delta(m-\epsilon)$
- $(\delta = 1 \implies \sum_{j\in N} a_j x_j \geq b) \iff \sum_{j\in N} a_j x_j \geq b + m(1-\delta)$
- $(\sum_{j\in N} a_j x_j \geq b \implies \delta = 1) \iff \sum_{j\in N} a_j x_j \leq b - \epsilon + \delta(M+\epsilon)$

1.7 Modelling Logical Constraints with Binary Variables

The following situations commonly occur in mathematical programming problems. We are given two constraints:

$$f(x_1, x_2, \dots, x_n) \leq 0, \qquad (1)$$
$$g(x_1, x_2, \dots, x_n) \leq 0, \qquad (2)$$

To generate the feasible region we are asked to ensure that:
(i) **Both-And** constraints: both (1) and (2) are satisfied;
(ii) **Either-Or** constraints: at least one of (1) and (2) is satisfied;
(iii) **If-Then** constraints: if (1) is satisfied then (2) must also be satisfied.

Both Constraints
This is the most common case in all examples that we have observed so far. The vast majority optimization software is designed to deal with such constraints.

Either-Or Constraints
Assume that we can find "big numbers" M_1 and M_2 (upper bounds for f and g

values), which ensures that $f(x_1, x_2, \dots, x_n) \le M_1$ and $g(x_1, x_2, \dots, x_n) \le M_2$ are satisfied for all feasible *x*.
Introduce binary variables

$$y_i = \begin{cases} 1, & if\ constraint\quad f_i(x_1, x_2, \dots, x_n) \le 0\quad is\ imposed\ to\ the\ model \\ 0, & otherwise\quad (f(x_1, x_2, \dots, x_n) \le M_i) \end{cases}, \ i=1, 2.$$

To guarantee at least one of them happens, we need $y_1 + y_2 \ge 1$.

Now the problem is how to relate conditions for the binary variables to original constraints using mathematical terms.

The following constraints work:

$$f(x_1, x_2, \dots, x_n) \le M_1(1 - y_1)$$
$$g(x_1, x_2, \dots, x_n) \le M_2(1 - y_2)$$

If $y_1 = 1\ then\ f(x_1, x_2, \dots, x_n) \le 0$, if $y_1 = 0\ then\ f(x_1, x_2, \dots, x_n) \le M_1$
Therefore, we have translated the either-or constraint into a set of constraints

(Which have to be satisfied simultaneously):

$$\begin{cases} f(x_1, x_2, \dots, x_n) \le M_1(1 - y_1), \\ g(x_1, x_2, \dots, x_n) \le M_2\ (1 - y_2), \\ y_1 + y_2 \ge 1, \\ y_1, y_2 \in \{0, 1\} \end{cases}$$ (**inclusive** either-or)

Now consider the generalized situation where *m* constraints are given as:

$$f_j(x_1, x_2, \dots, x_n) \le 0, \qquad j = 1,2, \dots, m$$

The **either-or constraint** <u>*at least* k *of them have to be true*</u> is then written as

$$\begin{cases} f_j(x_1, x_2, \dots, x_n) \le M_j(1 - y_j), & \forall\ j = 1,2, \dots, m \\ \sum_{j=1}^{m} y_j \ge k, & (inclusive\ either - or) \\ y \in \{0, 1\}^m \end{cases}$$

Notice that in this example, we allow both constraints (1) and (2) to be satisfied at a time (even if we have made y1 + y2 = 1), which can be used only for **inclusive either-or constraints**.

If-Then Constraints

In many applications we want to ensure that constraint (1) implies (2).
To derive this, we need the **If-Then** condition:

$$f(x_1, x_2, \dots, x_n) \leq 0 \implies g(x_1, x_2, \dots, x_n) \leq 0$$

It is clear that above implication is equivalent to:

$$f(x_1, x_2, \dots, x_n) > 0 \ \vee \ g(x_1, x_2, \dots, x_n) \leq 0$$

Therefore, we translate the **If-Then** condition to **Either-Or** condition.

Assume that we can find "big (positive) numbers" m_1 and M_2, which ensures that $f(x_1, x_2, \dots, x_n) > -m_1$ and $g(x_1, x_2, \dots, x_n) \leq M_2$ are satisfied for all feasible x.

$$y_1 = \begin{cases} 1, & if \ \ f(x_1, x_2, \dots, x_n) > 0 \\ 0, & f(x_1, x_2, \dots, x_n) > -m_1 \end{cases}$$

$$y_2 = \begin{cases} 1, & if \ \ g(x_1, x_2, \dots, x_n) \leq 0 \\ 0, & g(x_1, x_2, \dots, x_n) \leq M_2 \end{cases}$$

We have following constraints work:

$$\begin{cases} f(x_1, x_2, \dots, x_n) > -m_1(1 - y_1), & \text{(i)} \\ g(x_1, x_2, \dots, x_n) \leq M_2 \ (1 - y_2), & (ii) \\ y_1 + y_2 \geq 1, & (iii) \\ y_1, y_2 \in \{0, 1\} \end{cases}$$

Observe that if $f(x_1, x_2, \dots, x_n) \leq 0$, then (i) can be satisfied only if $y_1 = 0$. Then (iii) implies $y_2 = 1$ and therefore (ii) implies $g(x_1, x_2, \dots, x_n) \leq 0$. However if $f(x_1, x_2, \dots, x_n) > 0$, then (i) implies $y_1 = 1$. Therefore the value of y_2 is unrestricted and $g(x_1, x_2, \dots, x_n) > 0$ and $g(x_1, x_2, \dots, x_n) \leq 0$ are both possible.

Exercise

1. Use Binary variable δ to indicate whether or not the constraint $2x_1 + 3x_2 \leq 1$ is satisfied ($x_1, x_2 \in [0,1]$).
2. **The Fixed Charge Problem**: Let x represent the quantity of a product to be manufactured at a marginal cost per unit of C_1. In addition to the marginal, If the product is manufactured at all there is a set-up cost of C_2. Represent the cost graphically and write down a linear IP to model the cost.

1.8. Modelling Piecewise Linear Functions with Binary Variables

A Piecewise linear function consists of several straight-line segments.

$$\begin{aligned} &\max \ f(x) + C^T X \\ &s.t. \quad Ax \leq b \\ &x \in [LB, UB] \end{aligned}$$

$$f(x) = \begin{cases} 2x_1, & 0 \leq x_1 \leq 10 \\ \frac{3}{2}x_1, & 10 \leq x_1 \leq 20 \end{cases}$$

The points where the slop of the piecewise linear function changes (or the range of definition of the function ends) are called the break points of the function. Thus, 0, 10, 30, 40 and 50 are the break points of the above function.

A piecewise linear function is not a linear function, so one might think that linear programming could not be used to solve optimization problems involving these functions. By using 0-1 variables, however, piecewise linear functions (and/or even some smooth nonlinear functions) can be represented (and/or approximated) in linear form.

Suppose that a piecewise linear function f(x) has break points $b_1, b_2, \ldots, b_n$. As we have $n-1$ segments, a point x locates (exclusively) either on one segment or the other.
Therefore, let

$$y_k = \begin{cases} 1, & if\ point\ x\ located\ on\ segment\ (k) \\ 0, & otherwise \end{cases}$$

$$k = 1, 2, \ldots, n-1$$

We must have:

$$\sum_{k=1}^{n-1} y_k = 1$$

For some k $(k = 1, 2, ..., n-1)$, $b_k \leq x \leq b_{k+1}$. Then, given $\lambda_k \in [0, 1]$, x may be written as

$$x = \lambda_k b_k + (1 - \lambda_k) b_{k+1}.$$

Because $f(x)$ is linear for $b_k \leq x \leq b_{k+1}$, we can write

$$f(x) = \lambda_k f(b_k) + (1 - \lambda_k) f(b_{k+1})$$

Replacing $(1 - \lambda_k)$ by λ_{k+1}, $\lambda_{k+1} \in [0, 1]$,

$$f(x) = \lambda_k f(b_k) + \lambda_{k+1} f(b_{k+1})$$

$$\lambda_k + \lambda_{k+1} = 1$$

Furthermore, if only segment k is selected, we can use the break points at the end of each side to generate the linear function. This means:

$$\lambda_1 \leq y_1$$

$$\lambda_2 \leq y_1 + y_2$$

$$\dots$$

$$\lambda_{n-1} \leq y_{n-2} + y_{n-1}$$

$$\lambda_n \leq y_{n-1}$$

Assemble all constraints we have

$$x = \sum_{k=1}^{n} \lambda_k b_k$$

$$\sum_{k=1}^{n-1} y_k = 1 \qquad \text{(one segment is selected)}$$

$$\sum_{k=1}^{n} \lambda_k = 1 \qquad \text{(linear combination of end-points)}$$

$$\lambda_1 \leq y_1$$

$$\lambda_2 \leq y_1 + y_2$$

$$\ldots$$

$$\lambda_n \leq y_{n-1}$$

$$y \in \{0,1\}^{n-1}$$

$$\lambda \in [0,1]^n$$

And wherever $f(x)$ occurs in the optimization problem, replace it by $\sum_{k=1}^{n} \lambda_k f(b_k)$.

Example

Consider the NLP problem

$$\begin{aligned} \min \quad & x_1^2 - 4x_1 - 2x_2 \\ \text{s.t.} \quad & x_1 + x_2 \leq 4 \\ & 2x_1 + x_2 \leq 5 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Its objective is nonlinear but separable. Let us see how to find a piecewise linear approximation to the nonlinear part, i.e. x_1^2.

Assume the grid over which this approximation will be carried out is $\{0, 1, 2, 2.5\}$.

First, replace the nonlinear term x_1^2 by the single linear term y. The grid points to be used are

$$b_1 = 0 : y(b_1) = b_1^2 = 0;$$

$$b_2 = 1 : y(b_2) = b_2^2 = 1;$$

$$b_3 = 2 : y(b_3) = b_3^2 = 4;$$

$$b_4 = 2.5 : y(b_4) = b_4^2 = 6.25.$$

Therefore, we can replace x_1^2 in the objective by $0\lambda_1 + 1\lambda_2 + 4\lambda_3 + 6.25\lambda_4$. And the problem becomes:

$$
\begin{array}{lll}
 & \min & 0\lambda_1 + 1\lambda_2 + 4\lambda_3 + 6.25\lambda_4 - 4x_1 - 2x_2 \\
x = \sum_{k=1}^{n} \lambda_k b_k & \Rightarrow & x_1 = 0\lambda_1 + 1\lambda_2 + 2\lambda_3 + 2.5\lambda_4 \\
\sum_{k=1}^{n-1} y_k = 1 & \Rightarrow & y_1 + y_2 + y_3 = 1 \\
\sum_{k=1}^{n} \lambda_k = 1 & \Rightarrow & \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1 \\
\lambda_1 \leq y_1 & \Rightarrow & \lambda_1 \leq y_1 \\
\lambda_2 \leq y_1 + y_2 & \Rightarrow & \lambda_2 \leq y_1 + y_2 \\
\ldots & \Rightarrow & \lambda_3 \leq y_2 + y_3 \\
\lambda_n \leq y_{n-1} & \Rightarrow & \lambda_4 \leq y_3 \\
y \in \{0,1\}^{n-1} & \Rightarrow & y_1, y_2, y_3 \in \{0,1\} \\
\lambda \in [0,1]^n & \Rightarrow & \lambda_1, \lambda_2, \lambda_3, \lambda_4 \in [0,1] \\
 & & x_1 + x_2 \leq 4 \\
 & & 2x_1 + x_2 \leq 5 \\
 & & x_1, x_2 \geq 0
\end{array}
$$

RANSFORM NONBINARY TO 0-1 VARIABLE:

In this context, we define a general integer variable that can take on consecutive integer values between 0 and infinity. In this section, we refer to a discrete variable as one that takes on nonconsecutive integer values. For example, $z \in Z$={1, 5, 9, 21}. Both general integer variables and discrete variables are called non binary integer variables. Some algorithms apply only to problems with pure 0-1 variables

Transform Discrete Variable:

When a discrete variable is limited to take only one value in a given list, then the discrete variable can be expressed by a set of binary variables. For example, the discrete variable z may take on only one value in the set Z = {1, 5, 7, 9, 23}. To do this, we let a new 0-1 variable $y_i = 1$ $(i = 1, \ldots, 5)$ to represent the choice of the i-th element in the set and then add the following set of constraints to the problem:

$$z = 1y_1 + 5y_2 + 7y_3 + 9y_4 + 23y_5$$
$$y_1 + y_2 + y_3 + y_4 + y_5 = 1$$
$$y_i = 0 \text{ or } 1 \text{ for all } i$$

The above constraint equation, which is the sum of all of the binary variables equal to 1, is called a multiple choice constraint.