OPERATIONS RESEARCH II

LECTURE NOTES

(2026-2027)

Prof. Dr. Y. İlker Topcu & Prof. Dr. Özgür Kabak

Acknowledgements:

We would like to acknowledge Prof. W.L. Winston's "Operations Research: Applications and Algorithms“ (slides submitted by Brooks/Cole, a division of Thomson Learning, Inc.) and Prof. J.E. Beasley's lecture notes which greatly influence these notes...

We retain responsibility for all errors and would love to hear from readers...

CONTENTS

1. INTEGER PROGRAMMING – FORMULATING IPs

When formulating Linear Programs (LPs) we often found that, strictly, certain variables should have been regarded as taking integer values but, for the sake of convenience, we let them take fractional values reasoning that the variables were likely to be so large that any fractional part could be neglected.

While this is acceptable in some situations, in many cases it is not, and in such cases, we must find a numeric solution in which the variables take integer values.

Problems in which this is the case are called ***integer programs*** (***IP***'s) and the subject of solving such programs is called ***integer programming*** (also referred to by the initials ***IP***).

IP's occur frequently because many decisions are essentially discrete (such as yes/no, do/do not) in that one or more options must be chosen from a finite set of alternatives.

An IP in which all variables are required to be integers is called a ***pure IP*** problem.

If some variables are restricted to be integer and some are not then the problem is a ***mixed IP*** problem.

The case where the integer variables are restricted to be 0 or 1 comes up surprising often. Such problems are called ***pure (mixed) 0-1 programming*** problems or ***pure (mixed) binary IP*** problems.

For any IP we can generate an LP by taking the same objective function and same constraints but with the requirement that variables are integer replaced by appropriate continuous constraints:

"$x_i \geq 0$ and integer" can be replaced by $x_i \geq 0$

"$x_i = 0$ or 1" can be replaced by $x_i \geq 0$ and $x_i \leq 1$

The LP obtained by omitting all integer or 0-1 constraints on variables is called ***LP Relaxation of the IP*** (***LR***).

1.1 Budgeting Problems

Example 1. Capital Budgeting *(Winston 9.2, p. 478 – modified)*

Stock is considering four investments. Each investment yields a determined NPV ($8,000, $11,000, $6,000, $4,000). Each investment requires at certain cash flow at the present time ($5,000, $7,000, $4,000, $3,000). Currently Stock has $14,000 available for investment. Formulate an IP whose solution will tell Stock how to maximize the NPV obtained from the four investments.

Answer

Begin by defining a variable for each decision that Stockco must make.

In this case, we will use a 0-1 variable x_j for each investment:

If x_j is 1 then Stock will make investment *j*.

If it is 0, Stock will not make the investment.

This leads to the 0-1 programming problem:

$$\max z = 8x_1 + 11x_2 + 6x_3 + 4x_4$$

$$\text{s.t.} \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14$$

$$x_j = 0 \text{ or } 1 \quad (j = 1,2,3,4)$$

Comment

Now, a straightforward "bang for buck" (taking ratios of objective coefficient over constraint coefficient) suggests that investment 1 is the best choice.

Ignoring integrality constraints, the optimal linear programming solution is:

$x_1 = x_2 = 1$, $x_3 = 0.5$, and $x_4 = 0$ for a value of \$22K

Unfortunately, this solution is not integer. Rounding x_3 down to 0:

$x_1 = x_2 = 1$, $x_3 = x_4 = 0$ for a value of \$19K

There is a better integer solution (optimal solution):

$x_1 = 0$, $x_2 = x_3 = x_4 = 1$ for a value of \$21K

This example shows that rounding does not necessarily give an optimal value.

Example 2. Multiperiod Capital Budgeting

There are four possible projects, which each run for three years and have the following characteristics:

Which projects would you choose in order to maximize the total return?

		Capital requirements		
Project	Return	Year1	Year2	Year3
1	0.2	0.5	0.3	0.2
2	0.3	1	0.5	0.2
3	0.5	1.5	1.5	0.3
4	0.1	0.1	0.4	0.1
	Available capital	3.1	2.5	0.4

Answer

We will use a 0-1 variable x_j for each project:

x_j is 1 if we decide to do project *j*;

x_j is 0 otherwise (i.e. not do project *j*).

This leads to the 0-1 programming problem:

$$\max \quad 0.2\,x_1 + 0.3\,x_2 + 0.5\,x_3 + 0.1\,x_4$$

$$\text{s.t.} \quad 0.5\,x_1 + 1\,x_2 + 1.5\,x_3 + 0.1\,x_4 \leq 3.1$$

$$0.3\,x_1 + 0.8\,x_2 + 1.5\,x_3 + 0.4\,x_4 \leq 2.5$$

$0.2\ x_1 + 0.2\ x_2 + 0.3\ x_3 + 0.1\ x_4 \leq 0.4$

$x_j = 0$ or $1 \qquad j = 1, \ldots, 4$

Example 3. Capital Budgeting Extension

There are a number of additional constraints Stock might want to add.

Logical restrictions can be enforced using 0-1 variables:

Stock can only make two investments

$x_1 + x_2 + x_3 + x_4 \leq 2$

Any choice of three or four investments will have $x_1 + x_2 + x_3 + x_4 \geq 3$

If investment 2 is made, investment 4 must also be made

$x_2 \leq x_4$ or $x_2 - x_4 \leq 0$

If x_2 is 1, then x_4 is also 1; if x_2 is 0, then there is no restriction for x_4 (x_4 is 0 or 1)

If investment 1 is made, investment 3 cannot be made

$x_1 + x_3 \leq 1$

If x_1 is 1, then x_3 is 0; if x_1 is 0, then there is no restriction for x_3 (x_3 is 0 or 1)

Either investment 1 or investment 2 must be made (only one of them, not both)

$x_1 + x_2 = 1$

If x_1 is 1, then x_2 is 0 (only investment 1 is made); if x_1 is 0, then x_2 is 1 (only investment 2 is made)

1.2 Knapsack Problems

Any IP that has only one constraint is referred to as a knapsack problem.

Furthermore, the coefficients of this constraint and the objective are all non-negative.

The traditional story is that: There is a knapsack. There are a number of items, each with a size and a value. The objective is to maximize the total value of the items in the knapsack.

Knapsack problems are nice because they are (usually) easy to solve.

General representation of the knapsack problem:

max $z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$

s.t. $a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b$

$x_i = 0$ or 1 ($i = 1, 2, \ldots, n$)

where c_i is the value of item i, a_i is the weight of item i, and b is the total weight capacity.

For instance, the following is a knapsack problem:

Maximize $\quad 8\ x_1 + 11\ x_2 + 6\ x_3 + 4\ x_4$

Subject to $\quad 5\ x_1 + 7\ x_2 + 4\ x_3 + 3\ x_4 \leq 14$

$x_j = 0$ or $1 \qquad j = 1, \ldots 4$

1.3 Fixed Charge Problems

There is a cost associated with performing an activity at a nonzero level that does not depend on the level of the activity.

An important trick can be used to formulate many production and location problems involving the idea of a fixed charge as IP.

Example 4. Gandhi *(Winston 9.2, p. 480)*

Gandhi Co makes shirts, shorts, and pants using the limited labor and cloth described below. In addition, the machinery to make each product must be rented. Formulate an IP to maximize Gandhi's weekly profits.

	Shirts	**Shorts**	**Pants**	**Total Avail.**
Labor (hrs/wk)	3	2	6	150
Cloth (m^2/wk)	4	3	4	160
Rent for machine ($/wk)	200	150	100	
Variable unit cost	6	4	8	
Sale Price	12	8	15	

Answer

Let x_j be number of clothing produced (j = 1 (shirt), 2 (short), 3 (pants)).

Let y_j be 1 if any clothing *j* is manufactured and 0 otherwise (j = 1 (shirt), 2 (short), 3 (pants)).

Profit = Sales revenue – Variable Cost – Costs of renting machinery

For example, the profit from shirts is

$$z_1 = (12 - 6) x_1 - 200 y_1$$

Since supply of labor and cloth is limited, Gandhi faces two constraints.

To ensure $x_j > 0$ forces $y_j = 1$, we include the additional constraints

$$x_j \le M_j y_j$$

From the cloth constraint at most 40 shirts can be produced (M_1=40), so the additional constraint for shirts is not an additional limit on x_1 (If M_1 were not chosen large (say M_1=10), then the additional constraint for shirts would unnecessarily restrict the value of x_1).

From the cloth constraint at most 53 shorts can be produced (M_2=53)

From the labor constraint at most 25 pants can be produced (M_3=25)

We thus get the mixed (binary) integer problem:

$$\max \quad 6 x_1 + 4 x_2 + 7 x_3 - 200 y_1 - 150 y_2 - 100 y_3$$

$$\text{s.t.} \quad 3 x_1 + 2 x_2 + 6 x_3 \le 150 \qquad \text{(Labor constraint)}$$

$$4 x_1 + 3 x_2 + 4 x_3 \le 160 \qquad \text{(Cloth constraint)}$$

$$x_1 \le 40 y_1 \qquad \text{(Shirt production constraint)}$$

$$x_2 \le 53 y_2 \qquad \text{(Short production constraint)}$$

$$x_3 \le 25 y_3 \qquad \text{(Pant production constraint)}$$

$x_1, x_2, x_3 \geq 0$ and integer

$y_1, y_2, y_3 = 0$ or 1

Example 5. Warehouse problem

ATK-White are planning where to open warehouse (WH) and how to transport products from warehouses to costumers. 5 possible WH locations are considered to serve 4 customers. There is a fixed cost of opening and operating WHs (f_i). There is also transportation cost of transporting products from WHs to customers (c_{ij}). If a WH is opened, its capacity (K_i) cannot be exceeded. All demands of the customers (D_j) should be met.

Formulate an IP to minimize the total cost of ATK-White (WH opening costs and transportation costs) to meet the demands of the customers.

	Cust. 1	Cust. 2	Cust. 3	Cust. 4	Fixed cost (TL)	Capacity
WH -1	8TL	6TL	10TL	9TL	220,000	50,000
WH -2	9TL	12TL	13TL	7TL	280,000	60,000
WH -3	14TL	9TL	16TL	5TL	150,000	45,000
WH -4	18TL	16TL	10TL	9TL	290,000	80,000
WH -5	10TL	6TL	13TL	8TL	490,000	120,000
Demand	40,000	30,000	25,000	75,000		

Answer

Define decision variables:

x_{ij}: Amount of product transported from WH-i to Customer j. (i=1,2,3,4,5; j=1,2,3,4)

y_i: binary variable for opening WH-i. (i=1,2,3,4,5)

y_i= 1 if WH-i is opened, y_i= 0 if WH-i is not opened.

Objective is to minimize the sum of WH opening cost and total transportation cost.

$$Min\ Z = \sum_{i=1}^{5} f_i y_i + \sum_{i=1}^{5}\sum_{j=}^{4} c_{ij} x_{ij}$$

Subject to the following constraints:

$$\sum_{j=1}^{4} x_{ij} \leq K_i y_i \qquad \forall i$$

$$\sum_{i=1}^{5} x_{ij} \geq D_j \qquad \forall j$$

$$x_{ij} \geq 0 \quad \forall i, j, \qquad y_i \in \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \quad \forall i$$

1.4 Either-Or Constraints

Given two conditions

$f(x_1, x_2, \ldots, x_n) \leq 0$ (1)

$g(x_1, x_2, \ldots, x_n) \leq 0$ (2)

ensure that at least one is satisfied (1 or 2) by adding either-or-constraints:

$f(x_1, x_2, \ldots, x_n) \leq M\, y$

$g(x_1, x_2, \ldots, x_n) \leq M\,(1 - y)$

Here y is a 0-1 variable, and M is a number chosen large enough to ensure that both constraints are satisfied for all values of decision variables that satisfy the other constraints in the problem:

- If $y = 0$, then (1) and possibly (2) must be satisfied.
- If $y = 1$, then (2) and possibly (1) must be satisfied.

Example 1. Warehouse problem – Either-Or extension

Consider the Warehouse problem. Suppose that ATK-White has a chance of ignoring the demand of either customer 1 or customer 2. (i.e. they can choose to meet the demand of one of the two customers). How would you add this condition to the IP?

Answer

One of the following constraints should be satisfied:

$$x_{11} + x_{21} + x_{31} + x_{41} + x_{51} \geq 40{,}000$$
$$x_{12} + x_{22} + x_{32} + x_{42} + x_{52} \geq 30{,}000$$

Arrange the constraints to be in the format of Either-or constraint formulation:

$$-(x_{11} + x_{21} + x_{31} + x_{41} + x_{51}) + 40{,}000 \leq 0$$
$$-(x_{12} + x_{22} + x_{32} + x_{42} + x_{52}) + 30{,}000 \leq 0$$

In this way;

$$f(x) = -(x_{11} + x_{21} + x_{31} + x_{41} + x_{51}) + 40{,}000 \text{ and}$$
$$g(x) = -(x_{12} + x_{22} + x_{32} + x_{42} + x_{52}) + 30{,}000.$$

Add the following constraints to the model:

$$-(x_{11} + x_{21} + x_{31} + x_{41} + x_{51}) + 40000 \leq \mathrm{M}_1 y$$
$$-(x_{12} + x_{22} + x_{32} + x_{42} + x_{52}) + 30000 \leq \mathrm{M}_2(1 - y).$$

where $y \in \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$. M values can be considered as 40,000 and 30,000, respectively.

The final model would be as follows:

$$Min\ Z = \sum_{i=1}^{5} f_i y_i + \sum_{i=1}^{5} \sum_{j=}^{4} c_{ij} x_{ij}$$

Subject to

$$\sum_{j=1}^{4} x_{ij} \le K_i y_i \qquad \forall i$$

$$\sum_{i=1}^{5} x_{ij} \ge D_j \qquad j = 3,4,5.$$

$$-(x_{11} + x_{21} + x_{31} + x_{41} + x_{51}) + 40000 \le 40000y$$

$$-(x_{12} + x_{22} + x_{32} + x_{42} + x_{52}) + 30000 \le 30000(1 - y).$$

$$x_{ij} \ge 0 \quad \forall i, j, \qquad y_i \in \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \quad \forall i\,, \qquad y \in \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}.$$

Example 2. Compact Car

Suppose 1.5 tons of steel and 30 hours of labor are required for production of one compact car. At present, 6,000 tons of steel and 60,000 hours of labor are available. For an economically feasible production, at least 1,000 cars of compact car must be produced.

Answer

If it is infeasible then there will be no production. At least 1,000 cars must be produced or there will be no production.

- Given condition: $x = 0$ or $x \ge 1000$
- Sign restriction: $x \ge 0$ and Integer

$x = 0$ or $x \ge 1000$ → $x \le 0$ or $1000 - x \le 0$

For $f(x) = x$; $g(x) = 1000 - x$

We can replace the constraint by the following pair of linear constraints:

$x \le M\, y$

$1000 - x \le M\,(1 - y)$

$y = 0$ or 1

M = min (6,000/1.5, 60,000/30) = 2000

1.5 If-Then Constraints

Suppose we want to ensure that

a condition $f(x_1, x_2, \ldots, x_n) > 0$ implies

the condition $g(x_1, x_2, \ldots, x_n) \ge 0$

Then we include the following constraints in the formulation:

$-g(x_1, x_2, \ldots, x_n) \le M\, y$ (1)

$f(x_1, x_2, \ldots, x_n) \le M\,(1 - y)$ (2)

Here y is a 0-1 variable, and M is a large positive number, chosen large enough so that $f<M$ and $-g<M$ hold for all values of decision variables that satisfy the other constraints in the problem.

If $f > 0$, then (2) can be satisfied only if $y = 0$. (1) implies $-g \leq 0$ or $g \geq 0$, which is the desired result.

Example 3. Warehouse problem – If-then extension

Consider the Warehouse problem. How would you add the following condition to the model: If more than 40% of the capacity of WH-2 is used, WH-3 cannot be opened?

Answer

Initially, write a mathematical formulation for the condition:

If $x_{21} + x_{22} + x_{23} + x_{24} + x_{25} > 24{,}000$ then $y_3 = 0$.

$y_3 = 0 \rightarrow y_3 \leq 0 \rightarrow -y_3 \geq 0$

Arrange the constraints to be in the format of If-then constraint formulation

a constraint $f(x_1, x_2, \ldots, x_n) > 0$ implies the constraint $g(x_1, x_2, \ldots, x_n) \geq 0$

$$f(x) = x_{21} + x_{22} + x_{23} + x_{24} + x_{25} - 24{,}000 > 0$$

$$g(x) = -y_3$$

Add the following constraints to the model:

$$y_3 \leq \mathrm{M}_1 y$$

$$x_{21} + x_{22} + x_{23} + x_{24} + x_{25} - 24000 \leq \mathrm{M}_2(1 - y)$$

Where $y \in \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$. M values can be considered as 1 and 36,000, respectively.

The final model would be as follows:

$$Min\ Z = \sum_{i=1}^{5} f_i y_i + \sum_{i=1}^{5} \sum_{j=}^{4} c_{ij} x_{ij}$$

Subject to

$$\sum_{j=1}^{4} x_{ij} \leq K_i y_i \qquad \forall i$$

$$\sum_{i=1}^{5} x_{ij} \geq D_j \qquad \forall j$$

$$y_3 \leq y$$

$$x_{21} + x_{22} + x_{23} + x_{24} + x_{25} - 24000 \leq 36000(1 - y)$$

$$x_{ij} \geq 0 \quad \forall i, j, \qquad y_i \in \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \quad \forall i\,, \qquad y \in \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}.$$

1.6 Membership in Specified Subsets

Set covering, set packing, and set partitioning models are a special class of IP.

Using decision variables that equal 1 if an object is part of a solution and 0 otherwise, set covering, set packing, and set partitioning models formulate problems where the core issue is membership in specified subsets.

There are many applications in areas such as location (facility, fire/police station, warehouse), scheduling (crew, airline, truck, bus), delivery, vehicle routing, political districting, capital budgeting.

- ***Set covering*** problems arises when each set element must appear in ***at least*** one subset:

  $\sum_{j \in J} x_j \geq 1$

- ***Set packing*** problems arises when each set element must appear in ***at most*** one subset:

  $\sum_{j \in J} x_j \leq 1$

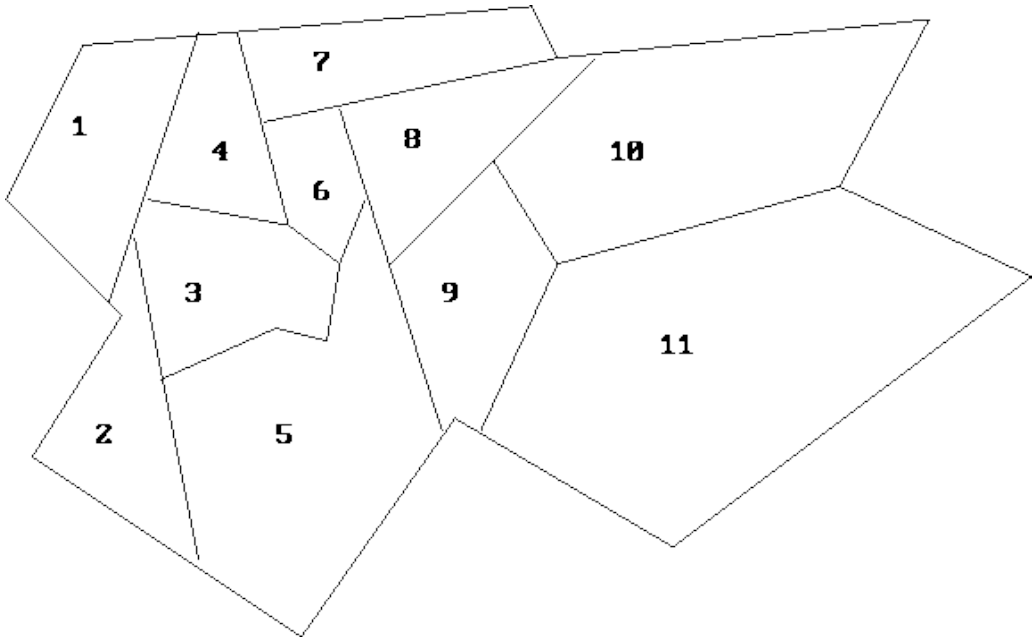
- ***Set partitioning*** problems arises when each set element must appear in ***exactly*** one subset:

  $\sum_{j \in J} x_j = 1$

Example 9. Fire Station

A county is reviewing the location of its fire stations.

The county is made up of a number of cities:



A fire station can be placed in any city. It is able to handle the fires for both its city and any adjacent city (any city with a non-zero border with its home city).

How many fire stations should be built and where?

Answer

We can create decision variable x_j for each city *j* (1 if we place a station in the city, 0 otherwise – j = 1,2,…,11):

Each constraint should state that there must be a station either in city *j* or in some adjacent city.

The jth column of the constraint matrix represents the set of cities that can be served by a fire station in city *j*.

We are asked to find a set of such subsets *j* that *covers* the set of all cities in the sense that every city appears in the service subset associated with *at least* one fire station.

There must be at least one fire station either in city j or in some adjacent city (set covering constraints).

$$\begin{aligned}
\min \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} \\
\text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 \geq 1 \text{ (city 1)} \\
& x_1 + x_2 + x_3 + x_5 \geq 1 \text{ (city 2)} \\
& x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 1 \text{ (city 3)} \\
& x_1 + x_3 + x_4 + x_6 + x_7 \geq 1 \text{ (city 4)} \\
& x_2 + x_3 + x_5 + x_6 + x_8 + x_9 \geq 1 \text{ (city 5)} \\
& x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \geq 1 \text{ (city 6)} \\
& x_4 + x_6 + x_7 + x_8 \geq 1 \text{ (city 7)} \\
& x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} \geq 1 \text{ (city 8)} \\
& x_5 + x_8 + x_9 + x_{10} + x_{11} \geq 1 \text{ (city 9)} \\
& x_8 + x_9 + x_{10} + x_{11} \geq 1 \text{ (city 10)} \\
& x_9 + x_{10} + x_{11} \geq 1 \text{ (city 11)}
\end{aligned}$$

All $x_j = 0$ or 1

Example 10. Timetable Scheduling

In an IE department, on Monday, 4 classes have to be assigned to classrooms. There are 3 classrooms devoted to the IE department. Each day is divided into 2 periods: morning and afternoon. Schedule a class timetable for the department.

Answer

x_{ijk} = 1 if class *i* is assigned to classroom *j* at time period *k*, 0 otherwise

i = 1, 2, 3, 4 (classes), *j* = 1, 2, 3 (rooms), *k* = 1, 2 (periods)

Each class should be assigned to exactly one classroom at a specific time period (set partitioning constraints):

$x_{111} + x_{112} + x_{121} + x_{122} + x_{131} + x_{132} = 1$ (class 1)

$x_{211} + x_{212} + x_{221} + x_{222} + x_{231} + x_{232} = 1$ (class 2)

$x_{311} + x_{312} + x_{321} + x_{322} + x_{331} + x_{332} = 1$ (class 3)

$x_{411} + x_{412} + x_{421} + x_{422} + x_{431} + x_{432} = 1$ (class 4)

At most one class can be held in each classroom at a specific time period (set packing constraints):

$x_{111} + x_{211} + x_{311} + x_{411} \leq 1$ (room 1 – morning)

$x_{112} + x_{212} + x_{312} + x_{412} \leq 1$ (room 1 – afternoon)

$x_{121} + x_{221} + x_{321} + x_{421} \leq 1$ (room 2 – morning)

$x_{122} + x_{222} + x_{322} + x_{422} \leq 1$ (room 2 – afternoon)

$x_{131} + x_{231} + x_{331} + x_{431} \leq 1$ (room 3 – morning)

$x_{132} + x_{232} + x_{332} + x_{432} \leq 1$ (room 3 – afternoon)

Therefore, the IP model that would be formulated to schedule the timetable will be:

max f (or any other objective function)

s.t. $x_{111} + x_{112} + x_{121} + x_{122} + x_{131} + x_{132} = 1$ (class 1)

$x_{211} + x_{212} + x_{221} + x_{222} + x_{231} + x_{232} = 1$ (class 2)

$x_{311} + x_{312} + x_{321} + x_{322} + x_{331} + x_{332} = 1$ (class 3)

$x_{411} + x_{412} + x_{421} + x_{422} + x_{431} + x_{432} = 1$ (class 4)

$x_{111} + x_{211} + x_{311} + x_{411} \leq 1$ (room 1 – morning)

$x_{112} + x_{212} + x_{312} + x_{412} \leq 1$ (room 1 – afternoon)

$x_{121} + x_{221} + x_{321} + x_{421} \leq 1$ (room 2 – morning)

$x_{122} + x_{222} + x_{322} + x_{422} \leq 1$ (room 2 – afternoon)

$x_{131} + x_{231} + x_{331} + x_{431} \leq 1$ (room 3 – morning)

$x_{132} + x_{232} + x_{332} + x_{432} \leq 1$ (room 3 – afternoon)

All $x_{ijk} = 0$ or 1

1.7 Traveling Salesperson Problems

"Given a number of cities and the costs of traveling from any city to any other city, what is the cheapest round-trip route (tour) that visits each city once and then returns to the starting city?" This problem is called the traveling salesperson problem (TSP), not surprisingly.

An itinerary that begins and ends at the same city and visits each city once is called a ***tour***.

Suppose there are *N* cities.

Let c_{ij} = Distance from city *i* to city *j* (for $i \neq j$) and

Let c_{ii} = M (a very large number relative to actual distances)

Also define x_{ij} as a 0-1 variable as follows:

x_{ij} = 1 if s/he goes from city *i* to city *j*;

x_{ij} = 0 otherwise

The formulation of the TSP is:

min $\sum_i \sum_j c_{ij} x_{ij}$

s.t. $\sum_i x_{ij} = 1$ for all j

$\sum_j x_{ij} = 1$ for all i

$u_i - u_j + N x_{ij} \leq N - 1$ for $i \neq j;\ \ i, j > 1$

All $x_{ij} = 0\ or\ 1$ All $u_i \geq 0$

The first set of constraints ensures that s/he arrives once at each city.

The second set of constraints ensures that s/he leaves each city once.

The third set of constraints ensure the following:

Any set of x_{ij}'s containing a subtour will be infeasible

Any set of x_{ij}'s that forms a tour will be feasible

$$u_i - u_j + N x_{ij} \leq N - 1 \qquad \text{for } i \neq j;\ \ i, j > 1$$

Assume N=5

Subtours: 1-5-2-1, 3-4-3 ???

Choose the subtour that does not contain city 1:

$$u_3 - u_4 + 5\, x_{34} \leq 4$$
$$u_4 - u_3 + 5\, x_{43} \leq 4$$
$$5\,(x_{34} + x_{43}) \leq 8$$

This rules out the possibility that $x_{34} = x_{43} = 1$

The formulation of an IP whose solution will solve a TSP becomes unwieldy and inefficient for large TSPs.

When using branch and bound methods to solve TSPs with many cities, large amounts of computer time may be required. For this reason, heuristics, which quickly lead to a good (but not necessarily optimal) solution to a TSP, are often used.

1.8 Piecewise Linear Function

0-1 variables can be used to model optimization problems involving piecewise linear functions.

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

The points where the slope of the piecewise linear function changes are called the break points of the function.

A piecewise linear function is not a linear function so linear programming can not be used to solve the optimization problem.

By using 0-1 variables, however, a piecewise linear function can be represented in linear form.

Suppose the piecewise linear function $f(x)$ has break points $b_1, b_2, \ldots, b_n$.

Step 1 Wherever $f(x)$ occurs in the optimization problem, replace $f(x)$ by

$$z_1 f(b_1) + z_2 f(b_2) + \ldots + z_n f(b_n).$$

Step 2 Add the following constraints to the problem:

$$z_1 \leq y_1,\ z_2 \leq y_1 + y_2,\ z_3 \leq y_2 + y_3,\ \ldots,\ z_{n-1} \leq y_{n-2} + y_{n-1},\ z_n \leq y_{n-1}$$

$$y_1 + y_2 + \ldots + y_{n-1} = 1$$

$$z_1 + z_2 + \ldots + z_n = 1$$

$$x = z_1 b_1 + z_2 b_2 + \ldots + z_n b_n$$

$y_i = 0$ or 1 (i=1,2,..., n-1); $z_i \geq 0$ (i=1,2,...,n)

Example 14. Euing Gas *(Winston 9.2, p. 492)*

Euing Gas produces two types of gasoline (gas 1 and gas 2) from two types of oil (oil 1 and oil 2). Each gallon of gas 1 must contain at least 50% of oil 1, and each gallon of gas 2 must contain at least 60% oil 1. Each gallon of gas 1 can be sold for 12¢, and each gallon of gas 2 can be sold for 14¢. Currently, 500 gallons of oil 1 and 1000 gallons of oil 2 are available. As many as 1500 more gallons of oil 1 can be purchased at the following prices: first 500 gallons, 25¢ per gallon; next 500 gallons, 20¢ per gallon; next 500 gallons, 15¢ per gallon.

Formulate an IP that will maximize Euing's profits (revenues – purchasing costs).

Answer

Except for the fact that the cost of purchasing additional oil 1 is a piecewise linear function, that is a straightforward blending problem:

x = amount of oil 1 purchased

x_{ij} = amount of oil i used to produce gas j (i,j=1,2)

Total revenue – cost of purchasing oil 1 =

$z = 12(x_{11} + x_{21}) + 14(x_{12} + x_{22}) - c(x)$

where

$$c(x) = \begin{cases} 25x & (0 \le x \le 500) \\ 20x + 2500 & (500 \le x \le 1000) \\ 15x + 7500 & (1000 \le x \le 1500) \end{cases}$$

Because $c(x)$ is a piecewise linear function, the objective function is not a linear function of x, and this optimization is not an LP.

However, we can transform this problem into an IP. After recalling that the break points for $c(x)$ are 0, 500, 1000, and 1500, we replace $c(x)$ and add additional constraints:

$c(x) = z_1\, c(0) + z_2\, c(500) + z_3\, c(1000) + z_4\, c(1500)$

$x = 0\, z_1 + 500\, z_2 + 1000\, z_3 + 1500\, z_4$

$z_1 \le y_1,\ z_2 \le y_1 + y_2,\ z_3 \le y_2 + y_3\ ,\ z_4 \le y_3$

$z_1 + z_2 + z_3 + z_4 = 1,$

$y_1 + y_2 + y_3 = 1$

$y_i = 0$ or 1 (i=1, 2, 3) ; $z_i \ge 0$ (i = 1, 2, 3, 4)

Problem Constraints:

Euing can use at most x + 500 gallons of oil 1:

$x_{11} + x_{12} \le x + 500$

Euing can use at most 1000 gallons of oil 2:

$x_{21} + x_{22} \le 1000$

The oil mixed to make gas 1 must be at least 50% oil 1:

$x_{11} / (x_{11} + x_{21}) \ge 0.5$ or $0.5x_{11} - 0.5x_{21} \ge 0$

The oil mixed to make gas 2 must be at least 60% oil 1:

$x_{12} / (x_{12} + x_{22}) \geq 0.6$ or $0.4x_{12} - 0.6x_{22} \geq 0$

Also, all variables must be nonnegative.

The final IP is as follows:

$$\max z = 12x_{11} + 12x_{21} + 14x_{12} + 14x_{22} - z_1c(0) - z_2c(500) - z_3c(1000) - z_4c(1500)$$

s.t. $x_{11} + x12 \leq x + 500$

$x_{21} + x22 \leq 1000$

$0.5x_{11} - 0.5x_{21} \geq 0$

$0.4x_{12} - 0.6x_{22} \geq 0$

$x = 0z_1 + 500z_2 + 1000z_3 + 1500z_4$

$z_1 \leq y_1$

$z_2 \leq y_1 + y_2$

$z_3 \leq y_2 + y_3$

$z_4 \leq y_3$

$y_1 + y_2 + y_3 = 1$

$z_1 + z_2 + z_3 + z_4 = 1$

$y_i = 0$ or 1 (i=1, 2, 3) ; $z_i \geq 0$ (i = 1, 2, 3, 4)

$x_{ij} \geq 0$

Report

The optimal solution to Euing's problem is

$z = 12{,}500$,

$x = 1000$, $x_{12} = 1500$, $x_{22} = 1000$, $y_2 = z_3 = 1$

Thus, Euing should purchase 1000 gallons of oil 1 and produce 2500 gallons of gas 2.

Total profit would be $125.

Review

To see why this formulation works, observe that

since $y_1 + y_2 + y_3 = 1$ and $y_i = 0$ or 1,

exactly one of the y_i's will equal 1, and the others will equal 0.

now, (1)-(7) imply that if $y_i = 1$, then z_i and z_{i+1} may be positive, but all the other z_i's must equal 0.

for instance, if $y_2 = 1$, then $y_1 = y_3 = 0$.

Then (2)-(5) become $z_1 \leq 0$, $z_2 \leq 1$, $z_3 \leq 1$ and $z_4 \leq 0$.

These constraints force $z_1 = z_4 = 0$ and allow z_2 and z_3 to be any nonnegative number less than or equal to 1.

We can now show that (1)-(7) correctly represent the piecewise linear function $c(x)$:

- Choose any value of x, say $x = 800$.
- Note that $b_2 = 500 \leq 800 \leq 1000 = b_3$.
- For $x = 800$, what value do our constraints assign to y_1, y_2 and y_3?
    - The value of $y_1 = 1$ is impossible, because if $y_1 = 1$, then $y_2 = y_3 = 0$. Then (4)-(5) force $z_3 = z_4 = 0$. Then (1) reduces to $800 = x = 500z_2$ which cannot be satisfied by $z_2 \leq 1$.

- Similarly, $y_3 = 1$ is impossible.
- If we try $y_2 = 1$, (2) and (5) force $z_1 = z_4 = 0$. Then (3) and (4) imply $z_2 \le 1$ and $z_3 \le 1$. Now (1) becomes $800 = x = 500z_2 + 1000z_3$. Because $z_2 + z_3 = 1$, we obtain $z_2 = 2/5$ and $z_3 = 3/5$.

- Now the objective function reduces to

  $12\ x_{11} + 12\ x_{21} + 14\ x_{12} + 14\ x_{22} - 2\ c(500) / 5 - 3\ c(1000) / 5$

  because $c(800) = 2\ c(500) / 5 + 3\ c(1000) / 5$

If a piecewise linear function $f(x)$ involved in a formulation has the property that the slope of the $f(x)$ becomes less favorable to the decision maker as x increases, then the tedious IP formulation is unnecessary.

Example 12. Dorian Auto *(Winston 9.2, p. 494)*

Dorian Auto has a $20,000 advertising budget. Dorian can purchase full page ads in two magazines: Inside Jocks (IJ) and Family Square (FS).

An exposure occurs when a person reads a Dorian Auto ad for the first time.

The number of exposures generated by each ad in IJ occurs as follows: ads 1-6, 10,000 exposures; ads 7-10, 3,000 exposures; ads 11-15, 2,500 exposures; ads 16+, 0 exposures.

For instance, 8 ads in IJ would generate 6 (10,000) + 2 (3,000) = 66,000 exposures.

The number of exposures generated by each ad in FS is as follows: ads 1-4, 8,000 exposures; 5-12, 6,000 exposures; ads 13-15, 2,000 exposures; ads 16+, 0 exposures.

Thus, 13 ads in FS would generate 4 (8,000) + 8 (6,000) + 1 (2,000) = 82,000 exposures.

Each full page ad in either magazine costs $1,000.

Assume there is no overlap in the readership of the two magazines.

Formulate an IP to maximize the number of exposures that Dorian can obtain with limited advertising funds.

Answer

If we define

x_1 = number of IJ ads yielding 10,000 exposures

x_2 = number of IJ ads yielding 3,000 exposures

x_3 = number of IJ ads yielding 2,500 exposures

y_1 = number of FS ads yielding 8,000 exposures

y_2 = number of FS ads yielding 6,000 exposures

y_3 = number of FS ads yielding 2,000 exposures

then the total number of exposures (in thousands) is given by

$10\ x_1 + 3\ x_2 + 2.5\ x_3 + 8\ y_1 + 6\ y_2 + 2\ y_3 = z$

Thus, Dorian wants to maximize z.

Because the total amount spent (in thousands) is just he total number of ads placed in both magazines, Dorian's budget constraint may be written as

$$x_1 + x_2 + x_3 + y_1 + y_2 + y_3 \le 20$$

The statement of the problem implies that

$$x_1 \le 6$$
$$x_2 \le 4$$
$$x_3 \le 5$$
$$y_1 \le 4$$
$$y_2 \le 8$$
$$y_3 \le 3$$

Adding the sign restrictions on each variable and noting that the each variable must be an integer, we obtain the following IP:

$$\max z = 10x_1 + 3x_2 + 2.5x_3 + 8y_1 + 6y_2 + 2y_3$$
$$\text{s.t.} \quad x_1 + x_2 + x_3 + y_1 + y_2 + y_3 \le 20$$
$$x_1 \le 6$$
$$x_2 \le 4$$
$$x_3 \le 5$$
$$y_1 \le 4$$
$$y_2 \le 8$$
$$y_3 \le 3$$
$$x_i, y_i \text{ integer } (i = 1, 2, 3)$$
$$x_i, y_i \ge 0 \ (i = 1, 2, 3)$$

Report

The optimal solution to Dorian's IP is

$$z = 146,\ x_1 = 6,\ x_2 = 2,\ y_1 = 4,\ y_2 = 8,\ x_3 = 0,\ y_3 = 0$$

Thus, in order to have possible maximum number of 146,000 exposures,

Dorian will place 8 ads in IJ and 12 ads in FS.

Review

Observe that the statement of the problem implies that x_2 cannot be positive unless x_1 assumes its maximum value of 6.

Because x_1 ads generate more exposures that x_2 ads, however, the act of maximizing ensures that x_2 will be positive only if x_1 has been made as large as possible.

Similarly, x_3 cannot be positive unless x_2 assumes its maximum value of 4.

Also, y_2 will be positive only if $y_1 = 4$, and y_3 will be positive only if $y_2 = 8$.

In this example, additional advertising in a magazine yielded diminishing returns.

This ensured that x_i (y_i) would be positive only if x_{i-1} (y_{i-1}) assumed its maximum value.

If additional advertising generated increasing returns, this formulation would not yield the correct solution.

Example 13. Modified Dorian Auto

Suppose that the number of exposures generated by each IJ ad was as follows: ads 1-6, 2,500 exposures; ads 7-10, 3,000 exposures; ads 11-15, 10,000 exposures; and that the number of exposures generated by each FS is as follows: ads 1-4, 2,000 exposures; ads 5-12, 6,000 exposures; ads 13-15; 8,000 exposures

Answer

If we define

x_1 = number of IJ ads generating 2,500 exposures

x_2 = number of IJ ads generating 3,000 exposures

x_3 = number of IJ ads generating 10,000 exposures

y_1 = number of FS ads generating 2,000 exposures

y_2 = number of FS ads generating 6,000 exposures

y_3 = number of FS ads generating 8,000 exposures

then the reasoning used in the previous example would lead to the following formulation:

$$\max z = 2.5x_1 + 3x_2 + 10x_3 + 2y_1 + 6y_2 + 8y_3$$

$$\begin{aligned} \text{s.t.} \quad & x_1 + x_2 + x_3 + y_1 + y_2 + y_3 \le 20 \\ & x_1 \le 6 \\ & x_2 \le 4 \\ & x_3 \le 5 \\ & y_1 \le 4 \\ & y_2 \le 8 \\ & y_3 \le 3 \end{aligned}$$

x_i, y_i integer (i = 1, 2, 3)

$x_i, y_i \ge 0$ (i = 1, 2, 3)

Report

The optimal solution to this IP is

$x_3 = 5, x_2 = 4, x_1 = 0, y_3 = 3, y_2 = 8, y_1 = 0$

which cannot be correct.

Therefore, we see that the type of formulation used in the Dorian Auto example is correct if the piecewise linear objective function has a less favorable slope for larger values of x.

In our second example (8'), the effectiveness of an ad increased as the number of ads in a magazine increased, and the act of maximizing will not ensure that x_i can be positive only if x_{i-1} assumes its maximum value.

In this case, the approach used in the Euing Gas example would yield a correct formulation.

Example 14. Machine tool design *(Based on Bazaraa et al. (2010) p.33)*

A lathe is used to reduce the diameter of a steel shaft whose length is 36 in. from 14 in. to 12 in. The speed x_1 (in revolutions per minute), the depth feed x_2 (in inches per minute), and the length feed x_3 (in inches per minute) must be determined. The duration of the cut is given by $36/x_2x_3$. The compression and side stresses exerted on the cutting tool are given by $30x_1 +$

$4500x_2$ and $40x_1 + 5000x_2 + 5000x_3$ pounds per square inch, respectively. The temperature (in degrees Fahrenheit) at the tip of the cutting tool is $200 + 0.5{x_1}^2 + 70{x_2}^3 + 150x_3$. The maximum compression stress, side stress, and temperature allowed are 150,000 psi, 100,000 psi, and 800°F, respectively. It is desired to determine the speed (which must be in the range from 600 rpm to 800 rpm), the depth feed, and the length feed such that the duration of the cut is minimized.

In order to use a linear model, the following approximation is made.

- Since $36/x_2x_3$ is minimized if and only if x_2x_3 is maximized, it was decided to replace the objective by the maximization of the minimum of x_2 and x_3.
- It is decided to convert non-linear function of the temperature to piecewise linear functions of x_1 and x_2.

Formulate the problem as a linear model and comment on the validity of the approximations used.

Answer

Maximization of the minimum of x_2 and x_3:

The objective would be $Max\ Min\ (x_2, x_3)$. To add this objective in a linear model, a new variable δ is defined, where $\delta = Min\ (x_2, x_3)$. And the following constraints are added in the model:

$$x_2 \geq \delta$$
$$x_3 \geq \delta$$

The objective function will be $Max\ \delta$.

Linearization of the nonlinear functions:

The temperature function the non-linear parts are separated as follows: $f_1 = 0.5{x_1}^2$, $f_2 = 70{x_2}^3$.

These functions (f_1 and f_2) are converted to piecewise linear function for an approximate modeling.

According to the given information in the question $600 \leq x_1 \leq 800$. This interval is divided in to 4 pieces as follows (For more accurate representation the interval can be divided into more pieces), and corresponding f_1 values are calculated:

x_1	**600**	**650**	**700**	**750**	**800**
f_1	180000	211250	245000	281250	320000

Then the following constraints are proposed for adding the piecewise linear function to the model:

$$f_1 = 180000\lambda_1 + 211250\lambda_2 + 245000\lambda_3 + 281250\lambda_4 + 320000\lambda_5$$
$$x_1 = 600\lambda_1 + 650\lambda_2 + 700\lambda_3 + 750\lambda_4 + 800\lambda_5$$

$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 = 1$$
$$\lambda_1 \leq z_1$$
$$\lambda_2 \leq z_1 + z_2$$
$$\lambda_3 \leq z_2 + z_3$$
$$\lambda_4 \leq z_3 + z_4$$
$$\lambda_5 \leq z_4$$
$$z_1 + z_2 + z_3 + z_4 = 1$$
$$z_1, z_2, z_3, z_4 \in \{0,1\}$$
$$\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5 \geq 0$$

According to the given information in the question, the maximum value of x_2 can be 2.252 (from the temperature function: $[800/70]^{1/3}$), and its minimum value is 0. So, $0 \leq x_2 \leq 2.26$. This interval is divided in to 4 pieces as follows and corresponding f_2 values are calculated:

x_2	0	0.565	1.130	3.390	2.26
f_2	0	12.63	101.00	2727.08	808.02

Then the following constraints are added as the piecewise linear function to the model:

$$f_2 = 0\alpha_1 + 12.63\alpha_2 + 101\alpha_3 + 2727.08\alpha_4 + 808.02\alpha_5$$
$$x_2 = 0\alpha_1 + 0.565 + 1.13\alpha_3 + 3.39\alpha_4 + 2.26\alpha_5$$
$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 = 1$$
$$\alpha_1 \leq k_1$$
$$\alpha_2 \leq k_1 + k_2$$
$$\alpha_3 \leq k_2 + k_3$$
$$\alpha_4 \leq k_3 + k_4$$
$$\alpha_5 \leq k_4$$
$$k_1 + k_2 + k_3 + k_4 = 1$$
$$k_1, k_2, k_3, k_4 \in \{0,1\}$$
$$\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5 \geq 0$$

There are also other constraints in the model:

The maximum compression stress: $30x_1 + 4500x_2 \leq 150000$

The maximum side stress: $40x_1 + 5000x_2 + 5000x_3 \leq 100000$

As a result, the following IP is developed:

$$Max\ \delta$$

Subject to

$$x_2 \geq \delta$$
$$x_3 \geq \delta$$
$$30x_1 + 4500x_2 \leq 150000$$

$$40x_1 + 5000x_2 + 5000x_3 \leq 100000$$

$$200 + (180000\lambda_1 + 211250\lambda_2 + 245000\lambda_3 + 281250\lambda_4 + 320000\lambda_5)$$
$$+ (0\alpha_1 + 12.63\alpha_2 + 101\alpha_3 + 2727.08\alpha_4 + 808.02\alpha_5) + 150x_3 \leq 800$$

$$x_1 = 600\lambda_1 + 650\lambda_2 + 700\lambda_3 + 750\lambda_4 + 800\lambda_5$$

$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 = 1$$

$$\lambda_1 \leq z_1$$

$$\lambda_2 \leq z_1 + z_2$$

$$\lambda_3 \leq z_2 + z_3$$

$$\lambda_4 \leq z_3 + z_4$$

$$\lambda_5 \leq z_4$$

$$z_1 + z_2 + z_3 + z_4 = 1$$

$$x_2 = 0\alpha_1 + 0.565 + 1.13\alpha_3 + 3.39\alpha_4 + 2.26\alpha_5$$

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 = 1$$

$$\alpha_1 \leq k_1$$

$$\alpha_2 \leq k_1 + k_2$$

$$\alpha_3 \leq k_2 + k_3$$

$$\alpha_4 \leq k_3 + k_4$$

$$\alpha_5 \leq k_4$$

$$k_1 + k_2 + k_3 + k_4 = 1$$

Sign restrictions and variable definitions:

$$\delta, x_1, x_2, x_3 \geq 0$$

$$z_1, z_2, z_3, z_4 \in \{0,1\}$$

$$\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5 \geq 0$$

$$k_1, k_2, k_3, k_4 \in \{0,1\}$$

$$\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5 \geq 0$$

2. INTEGER PROGRAMMING - SOLVING IPs

We have gone through a number of examples of IPs at the “Formulating IP Problems” section. “How can we get solutions to these models?” There are two common approaches:
The technique based on dividing the problem into a number of smaller problems in a *tree search* method called ***branch and bound***.
The method based on ***cutting planes*** (adding constraints to force integrality).
Actually, all these approaches involve solving a series of LP.
For solving LPs we have *general purpose* (independent of the LP being solved) and *computationally effective* (able to solve large LP's) algorithms (simplex or interior point). For solving IP's *no* similar general purpose and computationally effective algorithms exist.

2.1 Categorization

Categorization (w.r.t. Purpose)

- General purpose methods will solve any IP but potentially computationally ineffective (will only solve relatively small problems); or
- Special purpose methods are designed for one particular type of IP problem but potentially computationally more effective.

Categorization (w.r.t. Algorithm)

- Optimal algorithms mathematically guarantee to find the optimal solution
- Heuristic algorithms are used to solve a problem by trial and error when an optimal algorithm approach is impractical. They hopefully find a good feasible solution that, in objective function terms, is close to the optimal solution.

Why Heuristics?

Because the size of problem that we want to solve is beyond the computational limit of known optimal algorithms within the computer time we have available. We could solve optimally but feel that this is not worth the effort (time, money, etc) we would expend in finding the optimal solution. In fact, it is often the case that a well-designed heuristic algorithm can give good quality (near-optimal) results.

Solution Algorithms Categories

- General Purpose, Optimal
  Enumeration, branch and bound, cutting plane
- General Purpose, Heuristic
  Running a general-purpose optimal algorithm and terminating after a specified time
- Special Purpose, Optimal
  Tree search approaches based upon generating bounds via dual ascent, lagrangean relaxation

- Special Purpose, Heuristic
  Bound based heuristics, tabu search, simulated annealing, population heuristics (e.g. genetic algorithms), interchange

2.2 LP Relaxation

For any IP we can generate an LP by taking the same objective function and same constraints but with the requirement that variables are integer replaced by appropriate continuous constraints:

"x_i = 0 or 1" → x_i >= 0 and x_i <= 1

"$x_i \geq 0$ and integer" → x_i >= 0

The LP obtained by omitting all integer and 0-1 constraints on variables is called the ***LP Relaxation of the IP*** (***LR***). We can then solve this LR of the original IP.

Example 1. IP

Write the LR of the following IP:

Maximize $8\ x_1 + 5\ x_2$

Subject to $x_1 + x_2 \leq 6$

$9\ x_1 + 5\ x_2 \leq 45$

$x_1, x_2 \geq 0$ and integer

Answer

Maximize $8\ x_1 + 5\ x_2$

Subject to $x_1 + x_2 \leq 6$

$9\ x_1 + 5\ x_2 \leq 45$

$x_1, x_2 \geq 0$

Example 2. Binary IP

Write the LR of the following IP:

Maximize $0.2\ x_1 + 0.3\ x_2 + 0.5\ x_3 + 0.1\ x_4$

Subject to $0.5\ x_1 + 1\ x_2 + 1.5\ x_3 + 0.1\ x_4 \leq 3.1$

$0.3\ x_1 + 0.8\ x_2 + 1.5\ x_3 + 0.4\ x_4 \leq 2.5$

$0.2\ x_1 + 0.2\ x_2 + 0.3\ x_3 + 0.1\ x_4 \leq 0.4$

$x_j = 0$ or $1 \quad j = 1, \ldots 4$

Answer

Maximize $0.2\ x_1 + 0.3\ x_2 + 0.5\ x_3 + 0.1\ x_4$

Subject to $0.5\ x_1 + 1\ x_2 + 1.5\ x_3 + 0.1\ x_4 \leq 3.1$

$0.3\ x_1 + 0.8\ x_2 + 1.5\ x_3 + 0.4\ x_4 \leq 2.5$

$$0.2\,x_1 + 0.2\,x_2 + 0.3\,x_3 + 0.1\,x_4 \le 0.4$$

$$x_1 \le 1$$

$$x_2 \le 1$$

$$x_3 \le 1$$

$$x_4 \le 1$$

$$\text{All } x_i \ge 0$$

Example 3. Mixed IP

Write the LR of the following IP:

$$\max z = x_1 + x_2 + x_3$$

$$\text{s.t.} \quad x_1 + 6x_2 + x_3 \le 8$$

$$x_1 + 2x_2 + 1{,}5x_3 \le 2$$

$$2x_1 + x_2 + 5x_3 \le 8$$

$$x_1 \ge 0,\ x_2 \ge 0 \text{ and integer},\ x_3 \text{ binary}$$

Answer

$$\max z = x_1 + x_2 + x_3$$

$$\text{s.t.} \quad x_1 + 6x_2 + x_3 \le 8$$

$$x_1 + 2x_2 + 1{,}5x_3 \le 2$$

$$2x_1 + x_2 + 5x_3 \le 8$$

$$x_3 \le 1$$

$$x_1,\ x_2,\ x_3 \ge 0$$

Naturally Integer LP

If LR is optimized by integer variables then that solution is feasible and optimal for IP. In other words, if the solution is turned out to have all variables taking integer values at the optimal solution, it is also optimal solution for IP:

LR – IP Relation

Since LR is less constrained than IP:

- If IP is a maximization problem, the optimal objective value for LR is greater than or equal to that of IP.
- If IP is a minimization problem, the optimal objective value for LR is less than or equal to that of IP.
- If LR is infeasible, then so is IP.

So, solving LR does give some information. It gives a bound on the optimal value, and, if we are lucky, may give the optimal solution to IP.

2.3 Enumeration

Unlike LP (where variables took continuous values) in IP's (where all variables are integers) each variable can only take a finite number of discrete (integer) values.
Hence the obvious solution approach is simply to ***enumerate*** all these possibilities - calculating the value of the objective function at each one and choosing the (feasible) one with the optimal value.

Example 4. Multi-period Capital Budgeting

Maximize $0.2\,x_1 + 0.3\,x_2 + 0.5\,x_3 + 0.1\,x_4$

Subject to $0.5\,x_1 + 1\,x_2 + 1.5\,x_3 + 0.1\,x_4 \le 3.1$

$0.3\,x_1 + 0.8\,x_2 + 1.5\,x_3 + 0.4\,x_4 \le 2.5$

$0.2\,x_1 + 0.2\,x_2 + 0.3\,x_3 + 0.1\,x_4 \le 0.4$

$x_j = 0$ or $1 \qquad j = 1, \ldots 4$

Possible Solutions

There are 2^4=16 possible solutions:

0 0 0 0	do no projects	0 0 1 1	do two projects	1 1 1 0	do three projects
0 0 0 1	do one project	0 1 0 1		1 1 0 1	
0 0 1 0		1 0 0 1		1 0 1 1	
0 1 0 0		0 1 1 0		0 1 1 1	
1 0 0 0		1 0 1 0		1 1 1 1	do four projects
		1 1 0 0			

Review

Hence for our example, we merely have to examine 16 possibilities before we know precisely what the best possible solution is. This example illustrates a general truth about integer programming.
What makes solving the problem easy when it is small is precisely what makes it hard very quickly as the problem size increases.
This is simply illustrated: suppose we have 100 integer variables each with two possible integer values then there are 2x2x2x ... x2 = 2^{100} (approximately 10^{30}) possibilities which we have to enumerate (obviously many of these possibilities will be infeasible, but until we generate one, we cannot check it against the constraints to see if it is feasible).
For 100 integer variable - ***conceptually*** there is not a problem - simply enumerate all possibilities and choose the best one. But ***computationally*** (numerically) this is just impossible.

2.4 The Branch-and-Bound Method

The most effective general purpose optimal algorithm is an LP-based tree search approach called as ***branch and bound*** (B&B). The method was first put forward in the early 1960's by Land and Doig. This is *a way of systematically (implicitly) enumerating* feasible solutions such that the optimal integer solution is found.

Where this method differs from the enumeration method is that *not all* the feasible solutions are enumerated but only a part (hopefully a small part) of them. However, we can still *guarantee* that we will find the optimal integer solution. By solving a single subproblem, many possible solutions may be eliminated from consideration. Subproblems are generated by branching on an appropriately chosen fractional-valued variable.

Suppose that in a given subproblem (call it subp.1), assumes a fractional value between the integers *i* and *i*+1. Then the two newly generated subproblems:

Subp.2 = Subp.1 + Constraint "$x_i \geq i+1$"

Subp.3 = Subp.1 + Constraint "$x_i \leq I$"

If all variables have integer values in the optimal solution to the subproblem then the solution is a ***feasible*** solution for the original IP.

If the current feasible solution for the IP has a better *z*-value than any previously obtained feasible solution, then it becomes a ***candidate solution***, and its *z*-value becomes the current ***Lower Bound*** (***LB***) on the optimal *z*-value (for a max problem).

If it is unnecessary to branch on a subproblem, we say that it is fathomed (inactive):

- The subproblem is infeasible
- The subproblem yields an optimal solution in which all variables have integer values
- The optimal *z*-value for the subproblem does not exceed the current LB, so it cannot yield the optimal solution of the IP

Two general approaches are used to determine which subproblem should be solved next:

- Backtracking (LIFO)
  Leads us down one side of the B&B tree and finds a candidate solution. Then we backtrack our way up to the top of the other side of the tree.
- Jumptracking
  Solves all the problems created by branching. Then it branches again on the node with the best *z*-value. Often jumps from one side of the tree to the other.

A display of the subproblems that have been created is called a ***tree***.

Each subproblem is referred to as a ***node*** of the tree.

Each additional constraint is referred to as a line (***arc***) connecting two nodes (old subproblem and one of the new subproblems) of the tree.

2.5 B&B for Solving Pure IP Problems

Example 5. Pure IP *(Winston 9.3., p. 513)*

Solve the following IP using Branch and Bound method.

max $z = 8\ x_1 + 5\ x_2$

s.t. $x_1 + x_2 \le 6$

$9\ x_1 + 5\ x_2 \le 45$

$x_1, x_2 \ge 0$ and integer

Answer

Suppose that we were to solve the LR of the problem [replace "$x_1, x_2 \ge 0$ and integer" by "$x_1, x_2 \ge 0$"]. Then using any LP package or utilizing simplex or graphical solution method we get

$z = 165/4$, $x_1 = 15/4$, $x_2 = 9/4$

As a result of this we now know something about the optimal integer solution, namely that it is <= 165/4, i.e. this value of 165/4 is an ***Upper Bound*** on the optimal integer solution. This is because when we relax the integrality constraint we (as we are maximizing) end up with a solution value at least that of the optimal integer solution (and maybe better).

We arbitrarily choose a variable that is fractional in the optimal solution to the LR (subp.1): say x_1.

We need x_1 to be integer. We branch on x_1 and create two new subproblems:

Subp.2: LR + "$x_1 \ge 4$"

Subp.3: LR + "$x_1 \le 3$"

Observe that neither subp.2 nor subp.3 includes any points with $x_1 = 15/4$. This means that the optimal solution to LR can not recur when we solve these new subproblems.

We now arbitrarily choose to solve subp.2.

We see that the optimal solution to subp.2 is

$z = 41$, $x_1 = 4$, $x_2 = 9/5$

We choose x_2 that is fractional in the optimal solution to subp.2.

We need x_2 to be integer. We branch on x_2 and create two new subproblems:

Subp.4: LR + $x_1 \ge 4$ and $x_2 \ge 2$ = Subp.2 + $x_2 \ge 2$

Subp.5: LR + $x_1 \ge 4$ and $x_2 \le 1$ = Subp.2 + $x_2 \le 1$

The set of unsolved subproblems are consists of subp.3, 4, and 5.

We choose to solve the most recently created subproblem (This is called LIFO): The LIFO rule implies that we should next solve subp.4 or 5.

We now arbitrarily choose to solve subp.4.

We see that subp.4 is infeasible. Thus subp.4 cannot yield optimal solution to the IP.

Because any branches emanating from subp.4 will yield no useful information, it is fruitless to create them.

LIFO rule implies that we should next solve subp.5.

The optimal solution to subp.5 is

$z = 365/9$, $x_1 = 40/9$, $x_2 = 1$

We branch on fractional-valued x_1:

Subp.6: Subp.5 + $x_1 \geq 5$

Subp.7: Subp.5 + $x_1 \leq 4$

Subp.3, 6, and 7 are now unsolved.

The LIFO rule implies that we next solve subp.6 or 7.

We now arbitrarily choose to solve subp.7.

The optimal solution to subp.7 is

$z = 37$, $x_1 = 4$, $x_2 = 1$

As both variables assume integer values, this solution is ***feasible*** for the original IP → this solution is a ***candidate solution***

We must keep this candidate solution until a better feasible solution to the IP (if any exists) is found.

We may conclude that the optimal *z*-value for the IP ≥ 37 → ***Lower Bound*** (LB)

LIFO rule implies that we should next solve subp.6.

The optimal solution to subp.6 is

$z = 40$, $x_1 = 5$, $x_2 = 0$

It's *z*-value of 40 is larger than LB. Thus subp.7 cannot yield the optimal solution of the IP.

We update our LB to 40.

Subp.3 is the only remaining unsolved problem.

The optimal solution to subp.3 is

$z = 39$, $x_1 = 3$, $x_2 = 3$

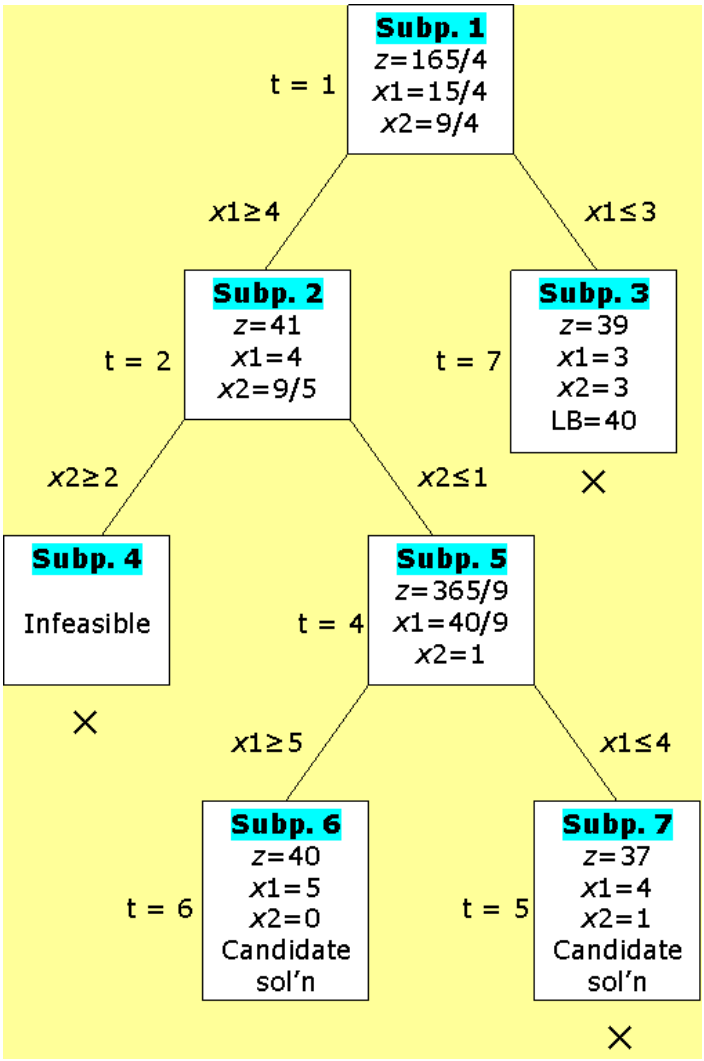
Subp.3 cannot yield a *z*-value exceeding the current LB, so it cannot yield the optimal solution to the IP.

Optimal Solution

Thus, the optimal solution to the IP

$z = 40$, $x_1 = 5$, $x_2 = 0$

Final B&B Tree



2.5.1 B&B for Solving Mixed IP Problems

In MIP, some variables are required to be integers and others are allowed to be either integer or non-integers. To solve a MIP by B&B method, modify the method by branching only on variables that are required to be integers. For a solution to a subproblem to be a candidate solution, it need only assign integer values to those variables that are required to be integers

Example 6. Mixed IP *(Winston 9.4., p. 523)*

Solve the following IP using Branch and Bound method.

max $z = 2\,x_1 + \;\; x_2$

s.t. $\quad 5\,x_1 + 2\,x_2 \le 8$

$\quad\quad x_1 + \;\; x_2 \le 3$

$\quad\quad x_1, x_2 \ge 0$; $x1$ integer

Answer

We solve the LR (subp.1) of the problem

[replace "$x_1 \geq 0$ and integer" by "$x_1 \geq 0$"]

Then using any LP package or utilizing simplex or graphical solution method we get

$z = 11/3$, $x_1 = 2/3$, $x_2 = 7/3$

Because x_2 is allowed to be fractional, we do not branch on x_2.

We branch on x_1 and create two new subproblems:

Subp.2: LR + $x_1 \geq 1$

Subp.3: LR + $x_1 \leq 0$

We see that the optimal solution to subp.2 is

$z = 7/2$, $x_1 = 1$, $x_2 = 3/2$

As only x_1 assume integer value, this solution is feasible for the original MIP → Candidate solution; LB = 7/2

The optimal solution to subp.3 is

$z = 3$, $x_1 = 0$, $x_2 = 3$

Subp.3 cannot yield a z-value exceeding the current LB, so it cannot yield the optimal solution to the MIP.

Optimal Solution

Thus, the optimal solution to the MIP

$z = 7/2$, $x_1 = 1$, $x_2 = 3/2$

2.5.2 B&B for Solving Binary IP Problems

One aspect of the B&B method greatly simplify:

Due to each variable equaling 0 or 1, branching on x_i will yield in

$x_i = 0$ and $x_i = 1$

Example 7. Binary IP

Solve the following IP using Branch and Bound method.

max z = $0.2 x_1 + 0.3 x_2 + 0.5 x_3 + 0.1 x_4$

s.t. $0.5 x_1 + 1 x_2 + 1.5 x_3 + 0.1 x_4 \leq 3.1$

$0.3 x_1 + 0.8 x_2 + 1.5 x_3 + 0.4 x_4 \leq 2.5$

$0.2 x_1 + 0.2 x_2 + 0.3 x_3 + 0.1 x_4 \leq 0.4$

$x_j = 0$ or $1 \quad j = 1, \ldots 4$

Answer

Replace "$x_j = 0$ or 1 (j=1,...,4)" by "$0 \leq x_j \leq 1$ (j=1,...,4)" → LR of the problem

Optimal solution to the LR:

z=0.65, x_2=0.5, x_3=1, x_1=x_4=0

The variable x_2 is fractional. To resolve this we can generate two new problems:

P1: LR + x_2=0

P2: LR + x_2=1

We now have two new subproblem to solve (*jumptracking*).

If we do this we get

P1 solution: z=0.6, x_1=0.5, x_3=1, x_2=x_4=0

P2 solution: z=0.63, x_2=1, x_3=0.67, x_1=x_4=0

Choosing subproblem P2 (the best z–value), we branch on x_3 and get

P3 (P2 + x_3=0) Solution: z=0.5, x_1=x_2=1, x_3=x_4=0

P4 (P2 + x_3=1) Solution: infeasible

P3 solution is feasible for the original binary IP → Candidate solution; LB = 0.5

Choosing the only remaining subproblem P1, we branch on x_1 and get

P5 (P1 + x_1=0) Solution: z=0.6, x_3=x_4=1, x_1=x_2=0

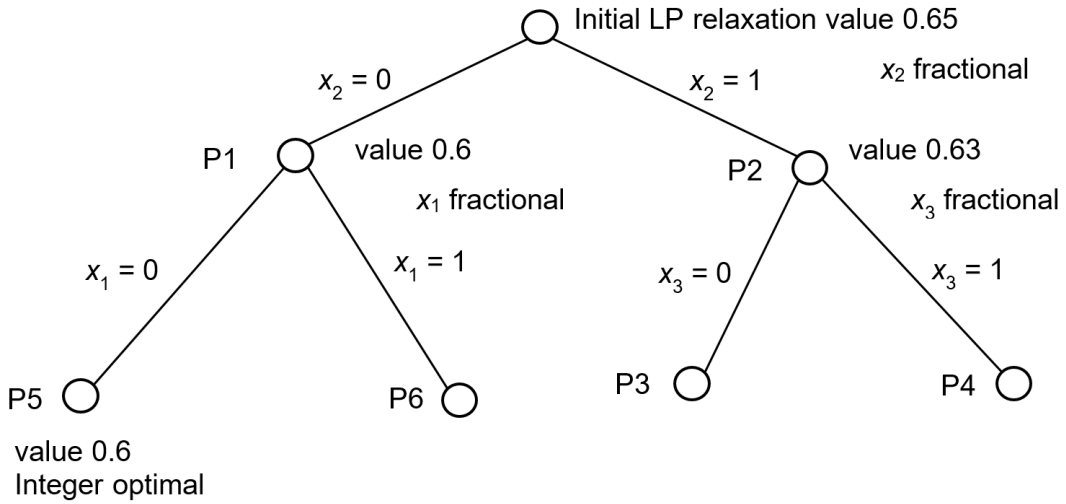
P6 (P1 + x_1=1) Solution: z=0.53, x_1=1, x_3=0.67, x_2=x_4=0

P5 solution is feasible for the original binary IP → New candidate solution; updated LB = 0.6

P6 cannot yield a z-value exceeding the current LB, so it cannot yield the optimal solution to the binary IP.

Thus, the optimal solution to the binary IP

z = 0.6, x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1



Review

Note here that B&B, like complete enumeration, also involves powers of 2 as we progress down the (binary) tree. However also note that we did not enumerate all possible integer solutions (of which there are 16). Instead, here we solved 7 LP's.

This is an important point, and indeed why tree search works at all. We do not need to examine as many LPs as there are possible solutions.

While the computational efficiency of tree search differs for different problems, it is this basic fact that enables us to solve problems that would be completely beyond us where we to try complete enumeration

2.5.3 B&B for Solving Knapsack Problems

Please recall that a knapsack problem is an IP, in which each variable must be equal to 0 or 1, with a single constraint:

max $z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$

s.t. $a_1x_1 + a_2x_2 + \cdots + a_nx_n \le b$

$x_i = 0$ or $1 \quad (i = 1, 2, \ldots, n)$

Two aspects of the B&B method greatly simplify:

- Due to each variable equaling 0 or 1, branching on x_i will yield in $x_i = 0$ and $x_i = 1$
- The LP relaxation may be solved by ***inspection*** instead of using any LP package or utilizing simplex or graphical solution method

Inspection

Recall that

c_i is the benefit obtained if item i is chosen

b is the total amount of an available resource

a_i is the amount of the available resource used by item i

Observe that ratio r_i (c_i/a_i) may be interpreted as the benefit item i earns for each unit of the resource used by item i. Thus, the best items have the largest value of r and the worst items have the smallest values of r.

To solve any subproblem resulting from a knapsack problem, compute all the ratios.

Then put the best item in the knapsack.

Then put the second-best item in the knapsack.

Continue in this fashion until the best remaining item will overfill the knapsack.

Then fill the knapsack with as much of this item as possible.

Example 8. Knapsack

Solve the following Knapsack problem using Branch and Bound method.

max $z =$ $8 x_1 + 11 x_2 + 6 x_3 + 4 x_4$

s.t. $5 x_1 + 7 x_2 + 4 x_3 + 3 x_4 \le 14$

$x_j = 0$ or $1 \qquad j = 1, \ldots 4$

Answer

We compute the ratios:

$r_1 = 8 / 5 = 1.6$

$r_2 = 11 / 7 = 1.57$

$r_3 = 6 / 4 = 1.5$

$r_4 = 4 / 3 = 1.33$

Using the ratios, LR solution is

$x_1 = 1, x_2 = 1, x_3 = 0.5, x_4 = 0, z = 22$

We branch on x_3 and get

P1 (LR + x_3=0) Solution: x_3=0, x_1=x_2=1, x_4=2/3, z=21.67

P2 (LR + x_3=1) Solution: x_3=x_1=1, x_2=5/7, x_4=0, z=21.85

Choosing subproblem P2 (the best z–value), we branch on x_2 and get

P3 (P2 + x_2=0) Solution: x_3=1, x_2=0, x_1=1, x_4=1, z=18

P4 (P2 + x_2=1) Solution: x_3=x_2=1, x_1=3/5, x_4=0, z=21.8

P3 solution is feasible for the original knapsack problem → Candidate solution; LB = 18

Choosing subproblem P4, we branch on x_1 and get

P5 (P4 + x_1=0) Solution: x_3=x_2=1, x_1=0, x_4=1, z=21

P6 (P4 + x_1=1) Solution: Infeasible (x_3=x_2=x_1=1: LHS=16)

P5 solution is feasible for the original knapsack problem → New candidate solution; updated LB = 21

The only remaining subproblem is P1 with solution value 21.67

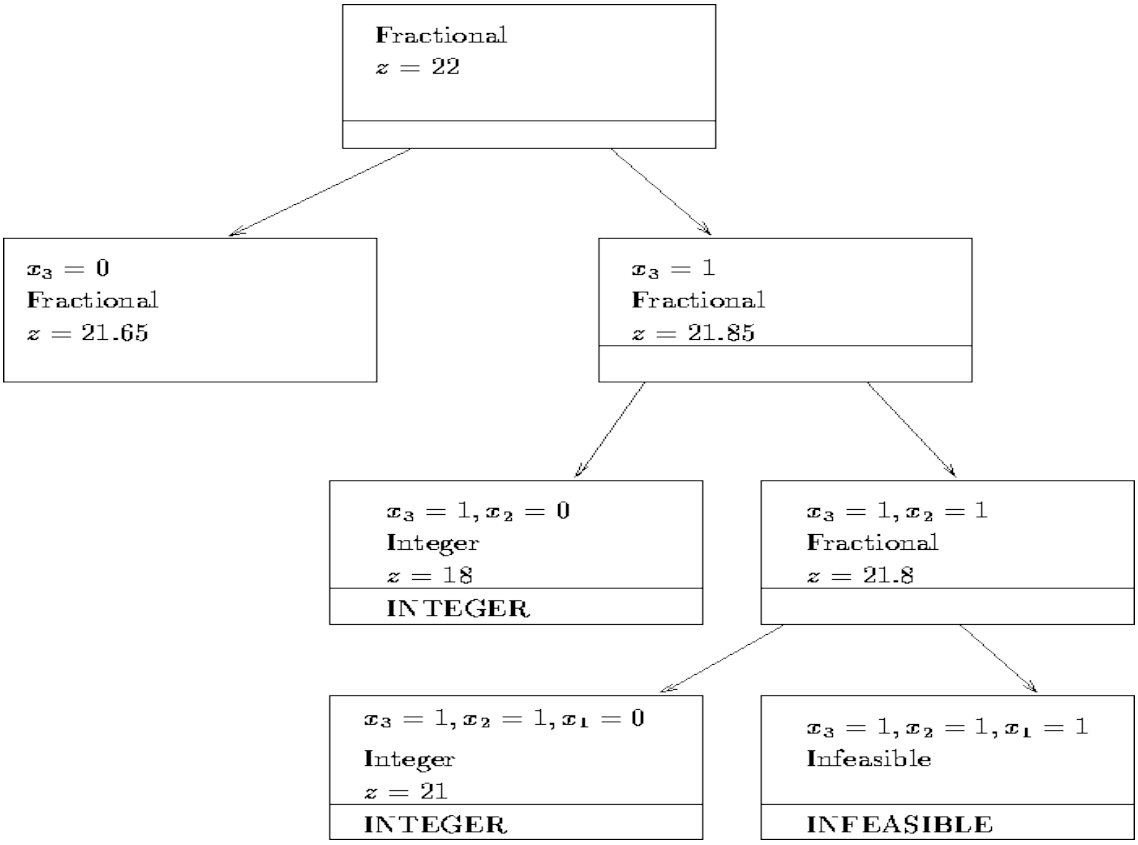
There is no better solution for this subproblem than 21. But we already have a solution with value 21.

It is not useful to search for another such solution. We can fathom P1 based on this bounding argument and mark P1 as inactive.

Optimal solution and Report

Thus, the optimal solution is z=21, x_1=0, x_2=1, x_3=1, x_4=1

Items 2, 3, and 4 should be put in the knapsack. In this case, the total value would be 21.



2.6 Combinatorial Optimization Problems

A ***combinatorial (discrete) optimization problem*** is any optimization problem that has a finite number of feasible solutions.

Examples of combinatorial optimization problems:

- Ten jobs must be processed on a single machine. It is known how long it takes to complete each job and the time at which each job must be completed (the job's due date). What ordering of the jobs minimizes the total delay of the 10 jobs?
- A salesperson must visit each of the 10 cities before returning to her/his home. What ordering of the cities minimizes the total distance the salesperson must travel before returning home? (TSP).

In each of these problems, many possible solutions must be considered.

A B&B approach is often an efficient way to solve them.

2.6.1 Job Shop Scheduling

Scheduling theory is one of the significant research areas in OR. It has been the subject of research with techniques ranging from simple dispatching rules to sophisticated learning algorithms. Job shop scheduling problem (JSP) is one of the most studied topics in scheduling theory which is known to be difficult to solve. Although the name job shop scheduling seems

to refer to an industrial problem, the structure of JSP is related to various applications, such as management, computing, and public services.

In JSP, there are n jobs planned to be scheduled on m machines. Each machine can process only one job and each job can be processed by only one machine at a given time (capacity constraints). The duration in which all operations for all jobs are completed is referred to as the makespan C_{max}. The scheduling costs may be the makespan, maximum lateness, maximum tardiness, maximum weighted completion time, total (weighted) completion time, and total (weighted) number of tardy jobs.

In **the single machine scheduling problem**, there are n jobs to be processed on a single machine. Each job has a processing time and a due date. The machine can process at most one job at a time. No preemption is allowed. The objective is to find a sequence of processing (i.e. an ordering of the jobs on the machine) to minimize the scheduling cost.

Example 9. Single Machine Scheduling

Please refer to Winston 9.6., p. 528

2.6.2 TSP

Please recall that

We define x_{ij} as a 0-1 variable:

x_{ij} = 1 if TS goes from city i to city j;

x_{ij} = 0 otherwise

c_{ij} = distance form city i to city j (for $i \neq j$)

c_{ii} = M (a very large number relative to actual distances)

An itinerary that begins and ends at the same city and visits each city once is called a tour.

It seems reasonable that we might be able to find the answer to TSP by solving an assignment problem having a cost matrix whose ijth is c_{ij}.

If the optimal solution to the assignment problem yields a tour, it is the optimal solution to the TSP. Unfortunately, the optimal solution to the assignment problem need not be a tour (may yield subtours).

If we could exclude all feasible solutions that contain subtours and then solve the assignment problem, we would obtain the optimal solution to TSP → Not easy to do...

Several B&B approaches have been developed for solving TSPs.

One approach starts with solving the preceding assignment problem (subproblem 1).

Because this subproblem contains no provisions to prevent subtours, it is a relaxation of the original TSP.

Thus, if the optimal solution to the subp.1 is feasible for the TSP (no subtours), then it is also optimal for the TSP.

If it is infeasible (contain subtours), we branch on the subp.1 in a way that will prevent one of subp.1's subtours from recurring in solutions to subsequent subproblems.

Example 10. TSP *(Winston 9.6., p. 530)*

Joe State lives in Gary, Indiana and owns insurance agencies in Gary, Fort Wayne, Evansville, Terre Haute, and South Bend.

Each December, he visits each of his insurance agencies.

The distance between each agency:

	(1)	**(2)**	**(3)**	**(4)**	**(5)**
Gary (1)	-	132	217	164	58
Fort Wayne (2)	132	-	290	201	79
Evansville (3)	217	290	-	113	303
Terre Haute (4)	164	201	113	-	196
South Bend (5)	58	79	303	196	-

What order of visiting his agencies will minimize the total distance traveled?

Answer

We first solve the assignment problem (subp.1) applying the Hungarian method to the cost matrix shown:

	(1)	**(2)**	**(3)**	**(4)**	**(5)**
(1)	M	132	217	164	58
(2)	132	M	290	201	79
(3)	217	290	M	113	303
(4)	164	201	113	M	196
(5)	58	79	303	196	M

The optimal solution will be:

$x_{15}=x_{21}=x_{34}=x_{43}=x_{52}=1$, $z=495$

The optimal solution to subp.1 contains two subtours:

- recommends going from Gary (1) to South Bend (5), then to Fort Wayne (2), and then back to Gary (1–5–2–1).
- also suggests that if Joe is in Evansville (3), he should go to Terre Haute (4) and then to Evansville (3–4–3).

Thus, the optimal solution cannot be the optimal solution to Joe's problem.

We arbitrarily choose to exclude the subtour 3-4-3.

Observe that the optimal solution to Joe's problem must have either $x_{34}=0$ or $x_{43}=0$.

Thus, we can branch on subp.1 by creating two new subproblems.

Subp.2: Subp.1 + ($x_{34}=0$, or $c_{34}=M$)

Subp.3: Subp.1 + ($x_{43}=0$, or $c_{43}=M$)

Now arbitrarily choose subp.2 to solve.

	(1)	(2)	(3)	(4)	(5)
(1)	M	132	217	164	58
(2)	132	M	290	201	79
(3)	217	290	M	**M**	303
(4)	164	201	113	M	196
(5)	58	79	303	196	M

The optimal solution will be:

$x_{14}=x_{25}=x_{31}=x_{43}=x_{52}=1$, $z=652$

This solution includes the subtours 1–4–3–1 and 2–5–2.

Thus, it cannot be the optimal solution to Joe's problem.

Following the LIFO approach, now branch subproblem 2 in an effort to exclude the subtour 2-5-2. Thus, we add two additional subproblems.

Subp.4: Subp.2 + ($x_{25}=0$, or $c_{25}=M$)

Subp.5: Subp.2 + ($x_{52}=0$, or $c_{52}=M$)

The assignment table for Subp.4:

	(1)	(2)	(3)	(4)	(5)
(1)	M	132	217	164	58
(2)	132	M	290	201	**M**
(3)	217	290	M	**M**	303
(4)	164	201	113	M	196
(5)	58	79	303	196	M

By using the Hungarian method on subp.4, we obtain the optimal solution

$x_{15}=x_{24}=x_{31}=x_{43}=x_{52}=1$, $z=668$

This solution contains no subtours and yields the tour 1–5–2–4–3–1

It is a candidate solution and any node that ***cannot*** yield a z-value < 668 may be eliminated from consideration.

We next solve subp.5.

$x_{14}=x_{43}=x_{32}=x_{25}=x_{51}=1$, $z=704$

This solution also yields a tour 1–4–3–2–5–1

But z=704 is not as good as the subp.4 candidate's z=668

Thus this subp.5 may be eliminated from consideration.

Only subp.3 remains.

The optimal solution

$x_{13}=x_{25}=x_{34}=x_{41}=x_{52}=1$, z =652.

This solution includes the subtours 1–3–4–1 and 2–5–2.

However, it is still possible for this subproblem to yield a solution with no subtours that beats z=668.

Next branch on subproblem 3 creating new subproblems.

Subp.6: Subp.3 + (x_{25}=0, or c_{25}=M)

Subp.7: Subp.3 + (x_{52}=0, or c_{52}=M)

Both of these subproblems have a z-value that is not smaller than 668.

Optimal solution and Report

Subp.4 thus yields the optimal solution:

$x_{15}=x_{24}=x_{31}=x_{43}=x_{52}=1$, z=668

Joe should travel from Gary (1) to South Bend (5), from South Bend to Fort Wayne (2), from Fort Wayne to Terre Haute (4), from Terre Haute to Evansville (3), and then back to Gary.

He will travel a total distance of 668 miles.

2.7 Heuristics for TSP

An IP formulation can be used to solve a TSP but can become unwieldy and inefficient for large TSPs. When using B&B methods to solve TSPs with many cities, large amounts of computer time is needed. Heuristic methods, or heuristics, can be used to quickly lead to a good (but not necessarily optimal) solution.

Two types of heuristic methods can be used to solve TSP:

- The Nearest-Neighbor
- The Cheapest-Insertion

Evaluation of Heuristics

Performance guarantees

Gives a worse-case bound on how far away from optimality a tour constructed by the heuristic can be

Probabilistic analysis

A heuristic is evaluated by assuming that the location of cities follows some known probability distribution

Empirical analysis

Heuristics are compared to the optimal solution for a number of problems for which the optimal tour is known

2.7.1 The Nearest-Neighbor Heuristic

STEPS:

1. Begin at any city and then "visit" the nearest city.
2. Then go to the unvisited city closest to the city we have most recently visited.
3. Continue in this fashion until a tour is obtained.
4. After applying this procedure starting at each city and reordering the resulting tours to begin at the hometown, recommend the best tour.

Example 11. Applying the NNH to TSP

We arbitrarily choose to begin at city 1.

Of the cities 2, 3, 4, and 5, city 5 is the closest city to city 1 → Generate the arc 1–5

Of the cities 2, 3, and 4, city 2 is the closest city to city 5 → 1–5–2

Of the cities 3 and 4, city 4 is the closest city to city 2 → 1–5–2–4

Joe must next visit city 3 and then return to city 1 → 1–5–2–4–3–1 (668 miles).

Begin at city 2:

Of the cities 1, 3, 4, and 5, city 5 is the closest city to city 2 → Generate the arc 2–5

Of the cities 1, 3, and 4, city 1 is the closest city to city 5 → 2–5–1

Of the cities 3 and 4, city 4 is the closest city to city 2 → 2–5–1–4

Joe must next visit city 3 and then return to city 2 → 2–5–1–4–3–2 (704 miles).

Begin at city 3: → 3-4-1-5-2-3 (704 miles)

Begin at city 4: → 4-3-1-5-2-4 (668 miles)

Begin at city 5: → 5-1-2-4-3-5 (807 miles)

Among the above-found tours, 668 miles that is found by beginning at city 1 and city 4 is the optimal solution. Notice that both tours are the same.

In this case, the NNH yields the optimal tour. However, for other problems it may not give the optimal solution.

2.7.2 The Cheapest-Insertion Heuristic

1. Begin at any city and find its closest neighbor.
2. Then create a subtour joining those two cities.
3. Next, replace an arc in the subtour (say, arc (*i, j*)) by the combinations of two arcs (*i*, *k*) and (*k*, *j*), where *k* is not in the current subtour that will increase the length of the subtour by the smallest (or cheapest) amount.
4. Continue with this procedure until a tour is obtained.

5. After applying this procedure starting at each city and reordering the resulting tours to begin at the hometown, recommend the best tour.

Example 12. Applying the CIH to TSP

Solution of Example 2.8 using CIH:

We arbitrarily choose to begin at city 1.

Of the cities 2, 3, 4, and 5, city 5 is the closest city to city 1 → Generate the arc 1–5

We create a subtour (1, 5)–(5, 1)

We could replace arc (1, 5) by (1, 2)–(2, 5), (1, 3)–(3, 5), or (1, 4)–(4, 5)

We could also replace (5, 1) by (5, 2)–(2, 1), (5, 3)–(3, 1), or (5, 4)–(4, 1)

The computations used to determine which arc of (1, 5)–(5, 1) should be replaced are given in the Table:

Arc replaced	Arcs added	Added length
(1, 5)*	(1, 2)–(2, 5)	$c_{12}+c_{25}-c_{15}=153$
(1, 5)	(1, 3)–(3, 5)	$c_{13}+c_{35}-c_{15}=462$
(1, 5)	(1, 4)–(4, 5)	$c_{14}+c_{45}-c_{15}=302$
(5, 1)*	(5, 2)–(2, 1)	$c_{52}+c_{21}-c_{51}=153$
(5, 1)	(5, 3)–(3, 1)	$c_{53}+c_{31}-c_{51}=462$
(5, 1)	(5, 4)–(4, 1)	$c_{54}+c_{41}-c_{51}=302$

* indicates the correct replacement: either (1, 5) or (5, 1)

We arbitrarily choose to replace arc (1, 5) by arcs (1, 2) and (2, 5) → New subtour: (1, 2)–(2, 5)–(5, 1)

We then determine which arc should be replaced

Arc replaced	Arcs added	Added length
(1, 2)	(1, 3)–(3, 2)	375
(1, 2)*	(1, 4)–(4, 2)	233
(2, 5)	(2, 3)–(3, 5)	514
(2, 5)	(2, 4)–(4, 5)	318
(5, 1)	(5, 3)–(3, 1)	462
(5, 1)	(5, 4)–(4, 1)	302

We now replace arc (1, 2) by arcs (1, 4) and (4, 2) → New subtour: (1, 4)–(4, 2)–(2, 5)–(5, 1)

Which arc should be replaced?

Arc replaced	Arcs added	Added length
(1, 4)*	(1, 3)–(3, 4)	166
(4, 2)	(4, 3)–(3, 2)	202
(2, 5)	(2, 3)–(3, 5)	514
(5, 1)	(5, 3)–(3, 1)	462

We now replace arc (1, 4) by arcs (1, 3) and (3, 4)

This yields the tour (1, 3)–(3, 4)–(4, 2)–(2, 5)–(5, 1)

In this case, the CIH yields the optimal tour.

But, in general, the CIH does not necessarily do so.

This procedure should be applied beginning at each city, and then the best tour found should be taken as solution.

2.8 Implicit Enumeration

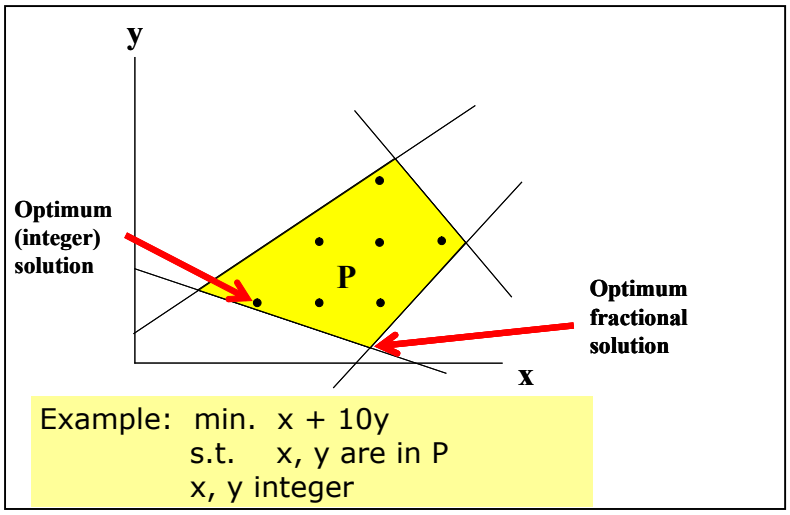
The method of implicit enumeration is often used to solve 0-1 IPs.
Implicit enumeration uses the fact that each variable must be equal to 0 or 1 to simplify both the branching and bounding components of the B&B process and to determine efficiently when a node is infeasible.
The tree used in the implicit enumeration method is similar to those used to solve 0-1 knapsack problems.
Some nodes have variable that are specified called ***fixed variables***.
All variables whose values are unspecified at a node are called ***free variables***.
For any node, a specification of the values of all the free variables is called a ***completion*** of the node.
Three main ideas used in implicit enumeration:

- Suppose we are at ay node with fixed variables, is there an easy way to find a good completion of the node that is feasible in the original 0-1 TSP?
- Even is the best completion of a node is not feasible, the best completion gives us a bound on the best objective function value that can be obtained via feasible completion of the node. This bound can be used to eliminate a node from consideration.
- At any node, is there an easy way to determine if all completions of the node are infeasible? In general, check whether a node has a feasible completion by looking at each constraint and assigning each free variable the best value for satisfying the constraint.

2.9 Cutting Planes

A linear inequality is a ***valid inequality*** for a given IP problem if it holds for all integer feasible solutions to the model.
Relaxations can often be strengthened dramatically by including valid inequalities that are not needed by a correct discrete model.
To strengthen a relaxation, a valid inequality must cut off (render infeasible) some feasible solutions to current LR that are not feasible in the IP model.
This need to cut off non-integer relaxation solutions is why valid inequalities are sometimes called ***cutting planes***.

Example 13. Cutting Plane - Conceptualization
Consider the following problem in the figure.

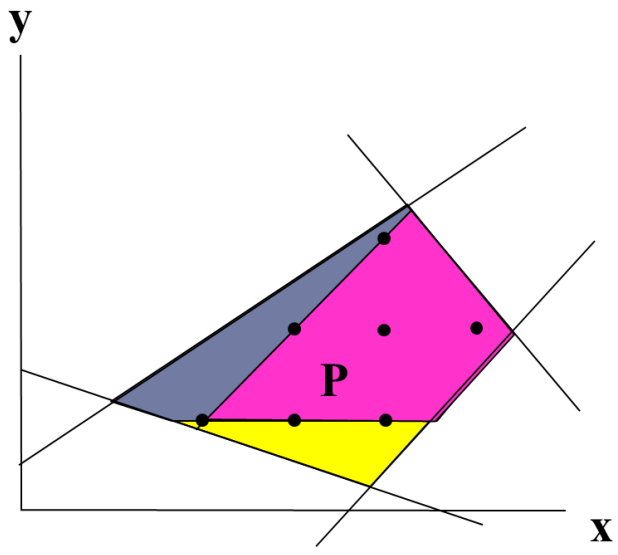


To solve this problem, the idea is to add constraints that eliminate fractional solutions to the LP without eliminating any integer solutions.

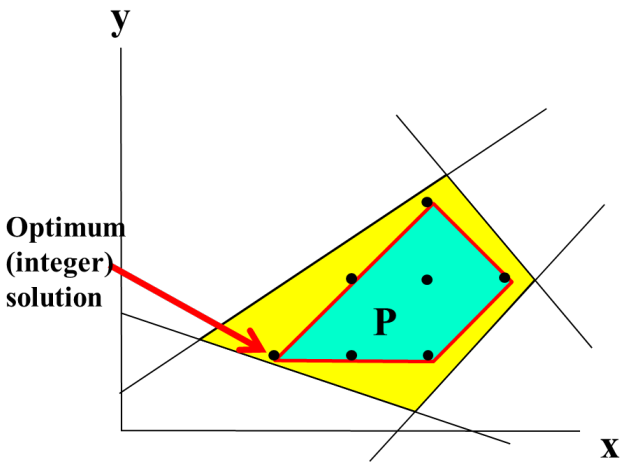
add "$y \geq 1$"

add "$y \leq x - 1$"

These constraints were obtained by inspection. We will develop techniques later.



If we add exactly the right inequalities, then every corner point of the LP will be integer, and the IP can be solved by solving the LP. We call this minimal LP, the ***convex hull*** of the IP solutions. For large problems, these constraints are hard to find. The tightest possible constraints are very useful, and are called ***facets***



Cut Classification

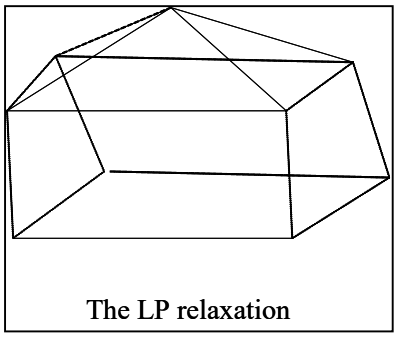
- General purpose

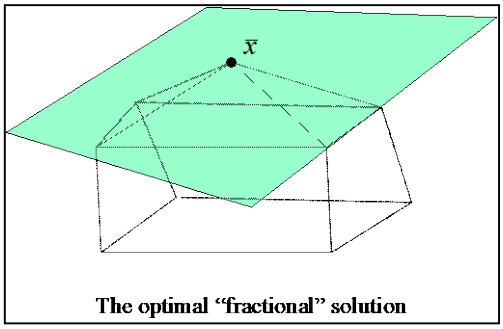
  A fractional extreme point can always be separated (LP-based approach, that works for IP)

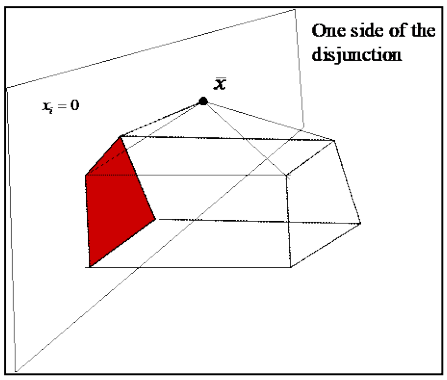
  - Disjunctive cuts
  - Gomory cutting planes
- Problem specific

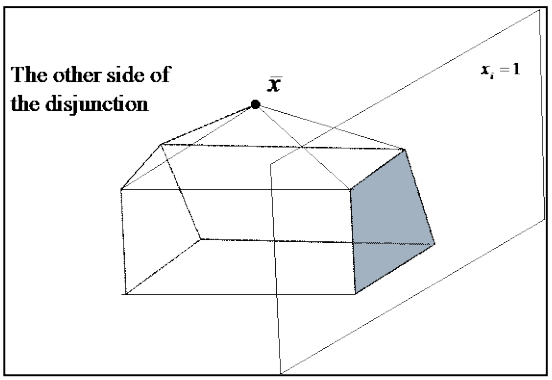
  Derived from problem structure, generally facets. (Knapsack, Set Packing...)

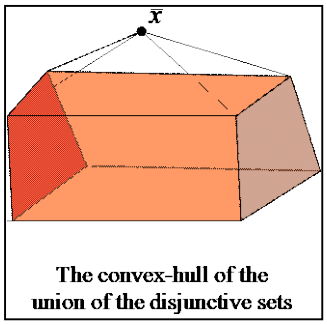
0-1 Disjunctive Cuts

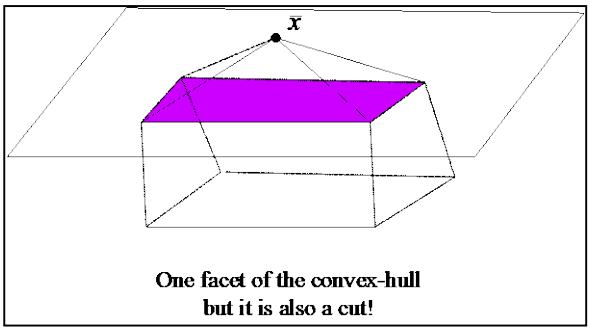


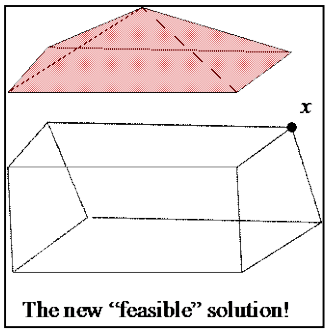












2.9.1 Cutting Plane Algorithm (Gomory cut)

Steps of the Gomory Cut Algorithm:

1. Find the optimal tableau for the IP's LR (LP Relaxation).
2. If all variables in the optimal solution assume integer values, we have found an optimal solution! Otherwise proceed to next step.
3. Pick a constraint in the optimal tableau whose RHS has the fractional part closest to ½.
4. For the constraint identified, put all of the integer parts on the left side (round down), and all the fractional parts on the right.
5. Generate the cut as:
   "RHS of the modified constraint" ≤ 0
6. Use the dual simplex to find the optimal solution to the LR, with the cut as an additional constraint.
7. If all variables assume integer values in the optimal solution, we have found an optimal solution to the IP.
8. Otherwise, proceed from step 3.

We continue this process until we obtain a solution in which all variables are integers. This will be an optimal solution to the IP.

Dual Simplex Method

Please recall that at dual simplex:

- We choose the most negative RHS.
- BV of this pivot row leaves the basis.

- For the variables that have a negative coefficient in the pivot row, we calculate the ratios (coefficient in R0 / coefficient in pivot row).
- Variable with the smallest ratio (absolute value) enters basis.

Example 14. Telfa Co. *(Winston 9.8., p. 546)*

Solve the following IP using Gomory Cut algorithm.

max z = 8 $x1$ + 5 $x2$

s.t. $x1$ + $x2 \leq 6$

9 $x1$ + 5 $x2 \leq 45$

$x1, x2 \geq 0$ and integer

Answer

If we ignore integrality, we get the following optimal tableau:

z	*x* 1	*x* 2	*s* 1	*s* 2	*RHS*
1	0	0	1.25	0.75	41.25
0	0	1	2.25	-0.25	2.25
0	1	0	-1.25	0.25	3.75

Let's choose the constraint whose RHS has the fractional part closest to ½ (Arbitrarily choose the second constraint):

$x_1 - 1.25\ s_1 + 0.25\ s_2 = 3.75$

We can manipulate this to put all of the integer parts on the left side (round down), and all the fractional parts on the right to get:

$x_1 - 2\ s_1 + 0\ s_2 - 3 = 0.75 - 0.75\ s_1 - 0.25\ s_2$

Now, note that the LHS consists only of integers, so the right-hand side must add up to an integer. It consists of some positive fraction minus a series of positive values. Therefore, the right-hand side cannot be a positive value. Therefore, we have derived the following constraint:

$0.75 - 0.75\ s_1 - 0.25\ s_2 \leq 0$

This constraint is satisfied by every feasible integer solution to our original problem. But, in our current solution, s_1 and s_2 both equal 0, which is infeasible to the above constraint. This means the above constraint is a cut, called the ***Gomory cut*** after its discoverer.

We can now add this constraint to the linear program and be guaranteed to find a different solution, one that might be integer.

z	*x* 1	*x* 2	***s* 1**	*s* 2	*s* 3	RHS
1	0	0	1.25	0.75	0	41.25
0	0	1	2.25	-0.25	0	2.25
0	1	0	-1.25	0.25	0	3.75
0	0	0	-0.75	-0.25	1	**-0.75**

The dual simplex ratio test indicates that s_1 should enter the basis instead of s_3.

The optimal solution is an IP solution:

$z = 40,\ x_1 = 5,\ x_2 = 0$

Example 15. Supplementary Problem

Solve the following IP using Gomory Cut algorithm.

$$\min z = 6x_1 + 8x_2$$

$$\text{s.t.} \quad 3x_1 + x_2 \geq 4$$

$$x_1 + 2x_2 \geq 4$$

$$x_1, x_2 \geq 0 \text{ and integer}$$

Answer

Optimal tableau for LR

z	*x* 1	*x* 2	*e* 1	*e* 2	*RHS*
1	0	0	-0.80	-3.60	17.60
0	1	0	-0.40	0.20	0.80
0	0	1	0.20	-0.60	1.60

Choose the second constraint

$$x_2 + 0.2\,e_1 - 0.6\,e_2 = 1.6$$

Manipulate this:

$$x_2 - e_2 - 1 = 0.6 - 0.2\,e_1 - 0.4\,e_2$$

Cut:

$$0.6 - 0.2\,e_1 - 0.4\,e_2 \leq 0$$

New LP tableau

z	*x* 1	*x* 2	***e* 1**	*e* 2	*s* 3	RHS
1	0	0	-0.8	-3.6	0	17.6
0	1	0	-0.4	0.2	0	0.8
0	0	1	0.2	-0.6	0	1.6
0	0	0	-0.2	-0.4	1	**-0.6**

The dual simplex ratio test indicates that e_1 should enter the basis instead of s_3.

The optimal solution is an IP solution:

$$z = 20,\ x_1 = 2,\ x_2 = 1$$

2.10 Branch & Cut

A variation of B&B termed ***Branch & Cut*** (**B&C**) takes advantage of valid inequalities to speed computation.

B&C algorithms modify the basic B&B strategy by attempting to strengthen relaxations with new inequalities before branching a partial solution.

Added constraints should hold for all feasible solutions to IP model, but they should cut off (render infeasible) the last relaxation optimum.

Example 16. Branch & Cut

Please refer to Rardin 12.5, p.676

2.11 Branch & Price

Branch & Price (**B&P**) is a generalization of LP based B&B specifically designed to handle IPs that contain a huge number of variables.

Columns are left out of the LR because there are too many to handle efficiently and most of them will have their associated variable equal to zero in an optimal solution anyway.

To check the optimality of an LP solution, a pricing problem is solved to try to identify columns with profitable reduced cost.

If profitable reduced cost columns are found, they are added and the LR is resolved.

If no profitable columns are found, the LP solution is optimal.

Branching occurs when the optimal LP solution does not satisfy the integrality conditions.

B&P applies column generation at every node of the B&B tree.

3. GOAL PROGRAMMING

In some situations, a decision maker may face multiple objectives, and there may be no point in an LP's feasible region satisfying all objectives.

In such a case, how can the decision maker choose a satisfactory decision?

Goal Programming, a variation of LP considering more than one objective (goals) in the objective function, is one technique that can be used in such situations.

Example 1. Priceler

The Leon Burnit Adveritsing Agency is trying to determine a TV advertising schedule for Priceler Auto Company.

Priceler has three goals:

- Its ads should be seen by at least 40 million high-income men (HIM).
- Its ads should be seen by at least 60 million low-income people (LIP).
- Its ads should be seen by at least 35 million high-income women (HIW).

Leon Burnit can purchase two types of ads: those shown during football games and those shown during soap operas.

At most, $600,000 can be spent on ads.

The advertising costs and potential audiences of a one-minute ad of each type are shown.

	Millions of Viewers			
Ad	**HIM**	**LIP**	**HIW**	**Cost (S)**
Football	7	10	5	100,000
Soap Opera	3	5	4	60,000

Leon Burnit must determine how many football ads and soap opera ads to purchase for Priceler.

Answer

Let x_1 = # of minutes of ads shown during football games

x_2 = # of minutes of ads shown during soap operas

Then any feasible solution to the following LP would meet Priceler's goals:

$$\min \text{ (or max) } z = 0x_1 + 0x_2$$

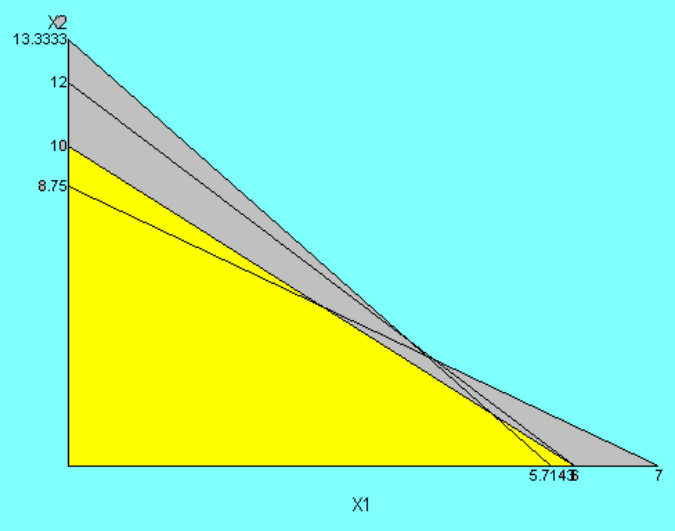
$$\text{s.t.} \quad 7x_1 + 3x_2 \geq 40 \quad \text{(HIM constraint)}$$

$$10x_1 + 5x_2 \geq 60 \quad \text{(LIP constraint)}$$

$$5x_1 + 4x_2 \geq 35 \quad \text{(HIW constraint)}$$

$$100x_1 + 60x_2 \leq 600 \quad \text{(Budget constraint)}$$

$$x_1, x_2 \geq 0$$



No point that satisfies the budget constraint meets all three of Priceler's goals → LP is infeasible

Since it is impossible to meet all of Priceler's goals, Burnit might ask Priceler to identify, for each goal, a cost that is incurred for failing to meet the goal.

Penalties

Each million exposures by which Priceler falls short of

- the HIM goal costs Priceler a $200,000 penalty
- the LIP goal costs Priceler a $100,000 penalty
- the HIW goal costs Priceler a $50,000 penalty

because of lost sales

Revised Answer

Burnit can then formulate an LP that minimizes the cost incurred in deviating from Priceler's three goals.

The trick is to transform each inequality constraint that represents one of Priceler's goals into an equality constraint.

Because it is not known whether a given solution will undersatisfy or oversatisfy a given goal, we need to define the following variables.

s_i^+ = amount by which the *i*th goal level is exceeded.

s_i^- = amount by which the *i*th goal level is underachieved

The s_i^+ and s_i^- are referred to as ***deviational variables***.

At least one or both deviational variables in a goal constraint must equal zero.

Burnit can minimize the penalty from Priceler's lost sales by solving the following LP.

$$\min \ z = 200s_1^- + 100s_2^- + 50s_3^-$$

$$\text{s.t.} \quad 7x_1 + 3x_2 + s_1^- - s_1^+ = 40 \ \text{(HIM constraint)}$$

$$10x_1 + 5x_2 + s_2^- - s_2^+ = 60 \ \text{(LIP constraint)}$$

$5x_1 + 4x_2 + s_3^- - s_3^+ = 35$ (HIW constraint)

$100x_1 + 60x_2 \leq 600$ (Budget constraint)

All variables nonnegative

Optimal Solution and Report

The optimal solution to this LP is

z=250, x_1=6, x_2=0,

s_1^+=2, s_2^+=0, s_3^+=0, s_1^-=0, s_2^-=0, s_3^-=5

6 minutes of ads should be shown during football games.

This meets goal 1 and goal 2 but fails to meet the least important goal (goal 3). 30 mio HIW sees Priceler's ads.

3.1 Deviational Variables

If failure to meet goal *i* occurs when the attained value of an attribute is numerically *smaller than the desired value* of goal *i*, then a term involving s_i^- will appear in the objective function.

If failure to meet goal *i* occurs when the attained value of an attribute is numerically *larger than the desired value* of goal *i*, then a term involving s_i^+ will appear in the objective function.

Also, if we want to *meet a goal exactly* and a penalty is assessed for going both over and under a goal, then terms involving s_i^-, s_i^+ will appear in the objective function.

Example 2. Modified Priceler

Suppose Priceler regards the budget restriction as a goal.

Priceler identifies a $1 penalty for each dollar by which this goal is unmet.

Answer

Then the appropriate goal programming formulation would be:

min $z = 200s_1^- + 100s_2^- + 50s_3^- + s_4^+$

s.t. $7x_1 + 3x_2 + s_1^- - s_1^+ = 40$ (HIM constraint)

$10x_1 + 5x_2 + s_2^- - s_2^+ = 60$ (LIP constraint)

$5x_1 + 4x_2 + s_3^- - s_3^+ = 35$ (HIW constraint)

$100x_1 + 60x_2 + s_4^- - s_4^+ = 600$ (Budget constraint)

All variables nonnegative

Optimal Solution & Report

The optimal solution to this LP is

z=100/3, x_1=13/3, x_2=10/3,

s_1^+=1/3, s_2^+=0, s_3^+=0, s_4^+=100/3, s_1^-=0, s_2^-=0, s_3^-=0, s_4^-=0

13/3 minutes of ads should be shown during football games and 10/3 minutes of ads should be shown during soap operas

This meets all three advertising goals by going $100/3 thousand over budget.

3.2 Preemptive Goal Programming

In many situations, a decision maker may not be able to determine precisely the relative importance of the goals.

When this is the case, ***preemptive goal programming*** (PGP) may prove to be a useful tool.

To apply PGP, the decision maker must rank her or his goals from the most important (goal 1) to least important (goal *n*).

The objective function coefficient P_i represents the weight of goal *i*:

$$P_1 >>> P_2 >>> \ldots >>> P_n$$

This definition ensures that the decision maker first tries to satisfy the most important goal.

Then among all points that satisfy goal 1, the decision maker tries to come as close as possible to satisfying goal 2, and so forth.

We continue in this fashion until the only way we can come closer to satisfying a goal is to increase the deviation from a higher priority goal.

3.2.1 Graphical Solution

When a PGP problem involves only two decision variables, the optimal solution can be found graphically.

Example 3. Beaver Creek Pottery *(Taylor, p. 358)*

Beaver Creek manufactures two products: Bowl, mug. Five workers (40 hours in total) process clay. They have 120 pounds of clay available for production.

	Labor	**Clay**	**Unit profit**
Bowl	1 hour	4 pounds	$40
Mug	2 hours	3 pounds	$50

If no other objective is stated, the problem can be formulated as follows:

maximize $z=40x_1 + 50x_2$

subject to

$$x_1 + 2x_2 \leq 40 \text{ hours of labor}$$
$$4x_1 + 3x_2 \leq 120 \text{ pounds of clay}$$
$$x_1, x_2 \geq 0$$

where x_1 = number of bowls produced

x_2 = number of mugs produced

Now suppose that the company states the following objectives (goals) *in order of importance.*

The company*:*

- does not want to use fewer than 40 hours of labor per day;
- would like to achieve a satisfactory profit level of $1,600 per day;
- prefers not to keep more than 120 pounds of clay on hand each day;
- would like to minimize the amount of overtime

Formulate and solve a goal programming model for the company.

Answer

Labor goal constraint (1, not less than 40 hours labor): minimize $P_1d_1^-$

Add profit goal constraint (2, achieve profit of \$1,600): minimize $P_1d_1^-$, $P_2d_2^-$

Add material goal constraint (3, avoid keeping more than 120 pounds of clay on hand): minimize $P_1d_1^-$, $P_2d_2^-$, $P_3d_3^+$

Labor goal constraint (4, minimum overtime): minimize $P_1d_1^-$, $P_2d_2^-$, $P_3d_3^+$, $P_4d_1^+$

Complete goal programming model:

minimize $P_1d_1^-$, $P_2d_2^-$, $P_3d_3^+$, $P_4d_1^+$

subject to

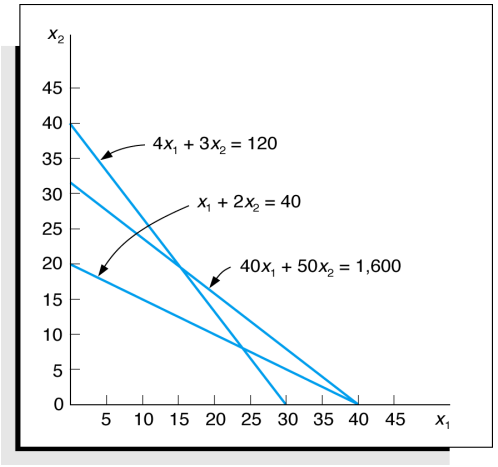
$$x_1 + 2x_2 + d_1^- - d_1^+ = 40$$

$$40x_1 + 50x_2 + d_2^- - d_2^+ = 1600$$

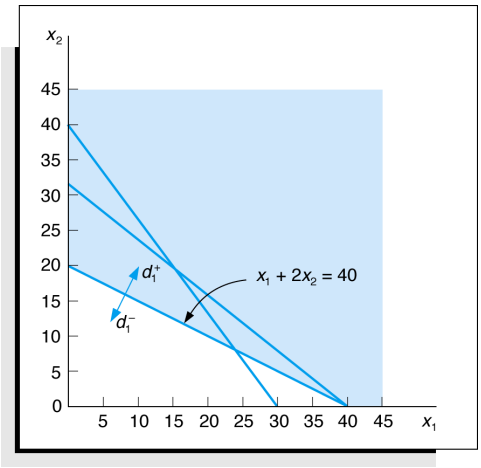
$$4x_1 + 3x_2 + d_3^- - d_3^+ = 120$$

$$x_1, x_2, d_1^-, d_1^+, d_2^-, d_2^+, d_3^-, d_3^+ \geq 0$$

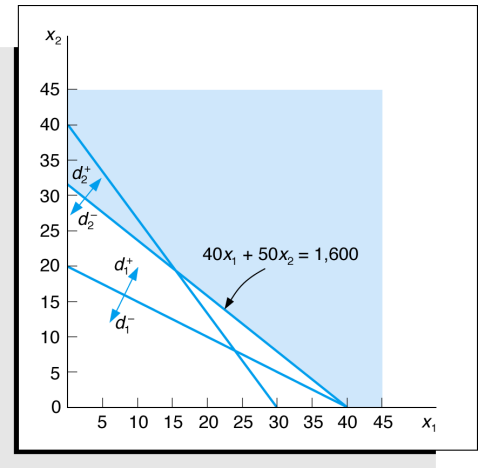
Goal constraints:



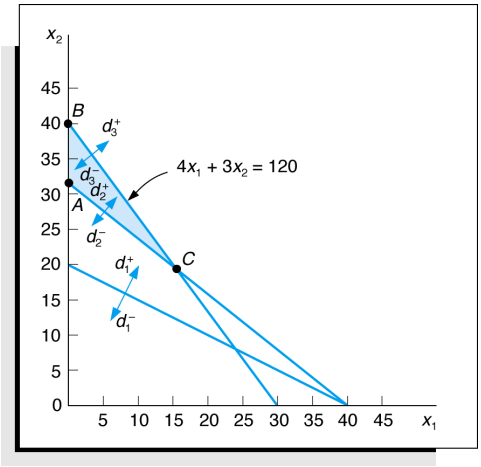
The first-priority goal: Minimize d_1^-

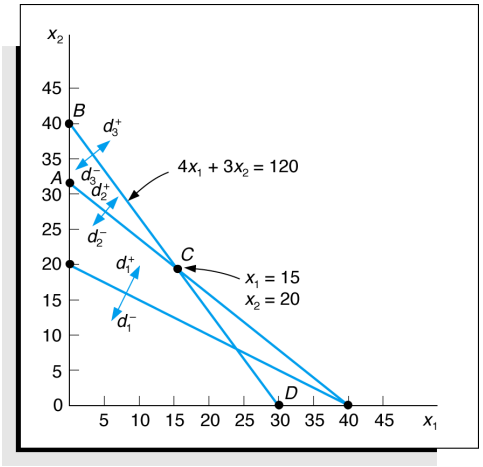


The second-priority goal: Minimize d_2^-



The third-priority goal: Minimize d_3^+





The fourth-priority goal: Minimize d_1^+

Optimal Solution & Report

$x_1 = 15$, $x_2 = 20$, $d_1^+ = 15$

15 bowls, 20 mugs should be produced.

This meets the first three goals by having 15 hours overtime labor.

3.2.2 Goal Programming Simplex

PGP problems can be solved by an extension of the simplex known as the ***goal programming simplex***.

The differences between the goal programming simplex and the ordinary simplex are:

- The ordinary simplex has a single row 0, whereas the goal programming simplex requires *n* row 0's (one for each goal).
- In the goal programming simplex, the entering variable is determined as follows (*PGP is a min problem*):
    - Find the highest priority goal (goal *i*') that has not been met
    - Find the variable with the most positive coefficient in row 0 (goal *i*') and enter this variable (subject to the following restriction) into the basis.
    - If, however, a variable has a negative coefficient in row 0 (goal *i*') associated with a goal having a higher priority than *i*', then the variable cannot enter the basis. In this case, try to find another variable with a positive coefficient in row 0 (goal *i*'). If no variable can enter the basis move on to row 0 (goal *i*'+1).
- When a pivot is performed, row 0 for each goal must be updated.
- A tableau will yield the optimal solution if all goals are satisfied, or if each variable that can enter the basis and reduce the value of z_i' for an unsatisfied goal *i*' will increase the deviation from some goal *i* having a higher priority than goal *i*'.

Example 4. Preemptive Priceler

Solve the following model using Goal Programming Simplex.

$$\min z = P_1 s_1^- + P_2 s_2^- + P_3 s_3^-$$

$$\text{s.t.} \quad 7x_1 + 3x_2 + s_1^- - s_1^+ = 40 \quad \text{(HIM constraint)}$$

$$10x_1 + 5x_2 + s_2^- - s_2^+ = 60 \quad \text{(LIP constraint)}$$

$$5x_1 + 4x_2 + s_3^- - s_3^+ = 35 \quad \text{(HIW constraint)}$$

$$100x_1 + 60x_2 \le 600 \quad \text{(Budget constraint)}$$

All variables nonnegative

Answer

We must separate the objective function into *n* components

$$\min z_1 = P_1 s_1^-$$

$$\min z_2 = P_2 s_2^-$$

$$\min z_3 = P_3 s_3^-$$

We have three Row 0's.

Row 0 (goal 1): $z_1 - P_1 s_1^- = 0$

Row 0 (goal 2): $z_2 - P_2 s_2^- = 0$

Row 0 (goal 3): $z_3 - P_3 s_3^- = 0$

We find that BV = $\{s_1^-, s_2^-, s_3^-, s_4\}$ (s_4 is the slack variable) is a starting bfs that could be used to solve Priceler problem

$$z_1 - P_1 s_1^- = 0$$

$$z_2 - P_2 s_2^- = 0$$

$$z_3 - P_3 s_3^- = 0$$

$$7x_1 + 3x_2 + s_1^- - s_1^+ = 40$$

$$10x_1 + 5x_2 + s_2^- - s_2^+ = 60$$

$$5x_1 + 4x_2 + s_3^- - s_3^+ = 35$$

$$100x_1 + 60x_2 + s_4 = 600$$

We must eliminate all variables in the starting basis from each row 0.

Add "P_1×Row 1" to "Row 0 (goal 1)"

Add "P_2×Row 2" to "Row 0 (goal 2)"

Add "P_3×Row 3" to "Row 0 (goal 3)"

	x_1	x_2	s_1^+	s_2^+	s_3^+	s_1^-	s_2^-	s_3^-	s_4	RHS	Ratio
Row 0 (HIM)	$7P_1$	$3P_1$	$-P_1$	0	0	0	0	0	0	$40P_1$	
Row 0 (LIP)	$10P_2$	$5P_2$	0	$-P_2$	0	0	0	0	0	$60P_2$	
Row 0 (HIW)	$5P_3$	$4P_3$	0	0	$-P_3$	0	0	0	0	$35P_3$	
HIM	7	3	-1	0	0	1	0	0	0	40	5.7143
LIP	10	5	0	-1	0	0	1	0	0	60	6
HIW	5	4	0	0	-1	0	0	1	0	35	7
Budget	100	60	0	0	0	0	0	0	1	600	6

	x_1	x_2	s_1^+	s_2^+	s_3^+	s_1^-	s_2^-	s_3^-	s_4	RHS	Ratio
Row 0 (HIM)	0	0	0	0	0	$-P_1$	0	0	0	0	
Row 0 (LIP)	0	$5P_2/7$	$10P_2/7$	$-P_2$	0	$-10P_2/7$	0	0	0	$20P_2/7$	
Row 0 (HIW)	0	$13P_3/7$	$5P_3/7$	0	$-P_3$	$-5P_3/7$	0	0	0	$45P_3/7$	
HIM	1	3/7	-1/7	0	0	1/7	0	0	0	40/7	
LIP	0	5/7	10/7	-1	0	-10/7	1	0	0	20/7	2
HIW	0	13/7	5/7	0	-1	-5/7	0	1	0	45/7	9
Budget	0	120/7	100/7	0	0	-100/7	0	0	1	200/7	2

	x_1	x_2	s_1^+	s_2^+	s_3^+	s_1^-	s_2^-	s_3^-	s_4	RHS
Row 0 (HIM)	0	0	0	0	0	$-P_1$	0	0	0	0
Row 0 (LIP)	0	$-P_2$	0	$-P_2$	0	0	0	0	$-P_2/10$	0
Row 0 (HIW)	0	P_3	0	0	$-P_3$	0	0	0	$-P_3/20$	$5P_3$
HIM	1	3/5	0	0	0	0	0	0	1/100	6
LIP	0	-1	0	-1	0	0	1	0	-1/10	0
HIW	0	1	0	0	-1	0	0	1	-1/20	5
Budget	0	6/5	1	0	0	-1	0	0	7/100	2

Optimal Solution & Report

The optimal solution to this LP is

$z_3=5P_3$, $x_1=6$, $x_2=0$,

$s_1^+=2$, $s_2^+=0$, $s_3^+=0$, $s_1^-=0$, $s_2^-=0$, $s_3^-=5$

6 minutes of ads should be shown during football games.

This meets goal 1 and goal 2 but fails to meet the least important goal (goal 3). 30 mio HIW sees Priceler's ads.

4. INTRODUCTION TO NONLINEAR PROGRAMMING

If the objective function and/or any constraint of a mathematical programming model is not linear then it is called Non-linear programming.

A general ***nonlinear program*** (***NLP***) can be expressed as follows:

Find the values of decision variables $x_1, x_2, \ldots x_n$ that

max (or min) $z = f(x_1, x_2, \ldots, x_n)$

s.t. $g_i(x_1, x_2, \ldots, x_n)$ (≤, =, or ≥) b_i $I = 1, 2, \ldots, m$

An LP is a special case of nonlinear programming!

As in LP, $f(x_1, x_2, \ldots, x_n)$ is the NLP's ***objective function***, and

$g_1(x_1, x_2, \ldots, x_n)$ (≤, =, or ≥)b_1,

$g_2(x_1, x_2, \ldots, x_n)$ (≤, =, or ≥)b_2,

…,

$g_m(x_1, x_2, \ldots, x_n)$ (≤, =, or ≥)b_m

are the NLP's ***constraints***.

An NLP is permitted to have non-linear objective and/or constraints.

An NLP with no constraints is an ***unconstrained NLP***.

The ***feasible region*** for NLP is the set of points $(x_1, x_2, \ldots, x_n)$ that satisfy the m constraints in the NLP.

If the NLP is a *maximization* problem, then any point x^* in the feasible region for which $f(x^*) \geq f(x)$ holds true for all points x in the feasible region is an ***optimal solution*** to the NLP.

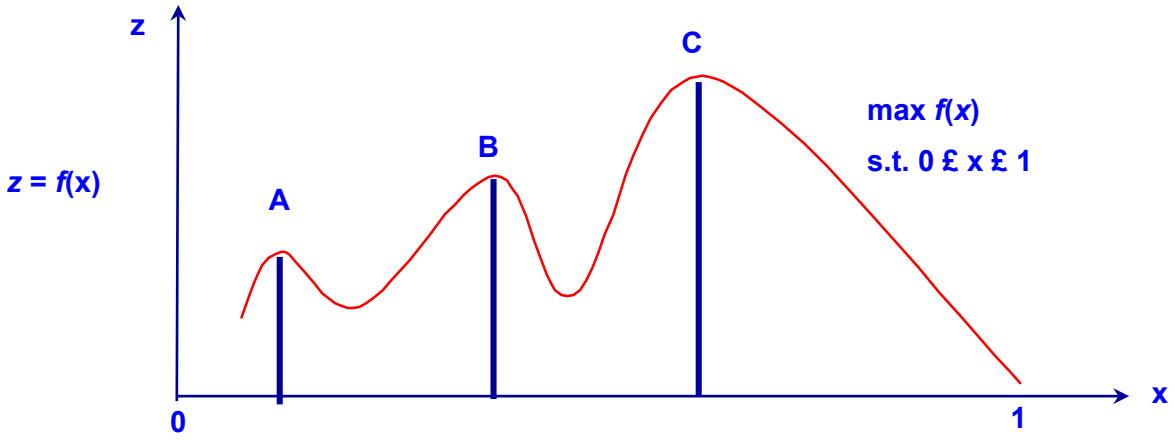
If the NLP is a *minimization* problem, then any point x^* in the feasible region for which $f(x^*) \leq f(x)$ holds true for all points x in the feasible region is an optimal solution to the NLP.

A point that is a local maximum or a local minimum is called a ***local extremum*** (or relative extremum)

Even if the feasible region for an NLP is a convex set, the optimal solution need not be an extreme point of the NLP's feasible region.

Let x = (x1, x2, ..., xn) be a feasible solution, then

x is a ***local maximum*** if f(x)≥f(x') for all feasible x' that are sufficiently close to x (i.e., xj-ε<xj'<xj+ε for all j and some small e).



There may be several locally optimal solutions: A, B, and C are all local maxima, but C is the unique optimal solution (***global maximum***) to NLP

4.1 Graphical Analysis

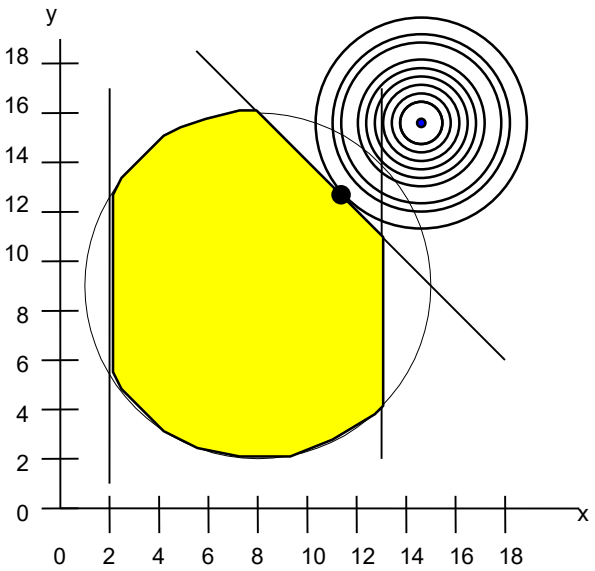
Similar to LP, NLPs with two decision variables can be solved graphically. Notice that it may not be possible to analyze all NLP problems bacuse of the complicated objective functions.

Example 4.1. Graphical Analysis

Solve the following NLP graphically.

$$\begin{aligned} \text{minimize} \quad & [(x - 14)^2 + (y - 15)^2]^{1/2} \\ \text{subject to} \quad & (x - 8)^2 + (y - 9)^2 \leq 49 \\ & x \geq 2 \\ & x \leq 13 \\ & x + y \leq 24 \end{aligned}$$

Answer



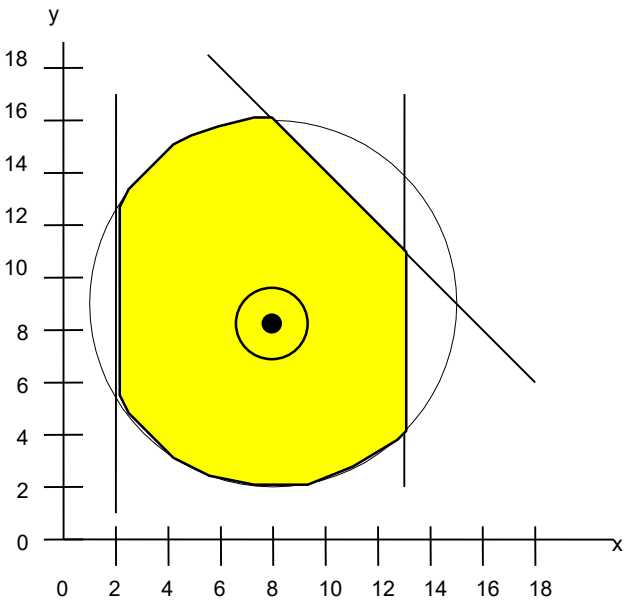
Notice that the optimal solution is not at a corner point. It is where the isocontour first hits the feasible region.

Example 4.2. Graphical Analysis

Solve the following NLP graphically.

$$\begin{aligned} \text{minimize} \quad & [(x-8)^2 + (y-8)^2]^{1/2} \\ \text{subject to} \quad & (x-8)^2 + (y-9)^2 \leq 49 \\ & x \geq 2 \\ & x \leq 13 \\ & x + y \leq 24 \end{aligned}$$

Answer



The optimal solution is not on the boundary of the feasible region.
It is inside the feasible region.

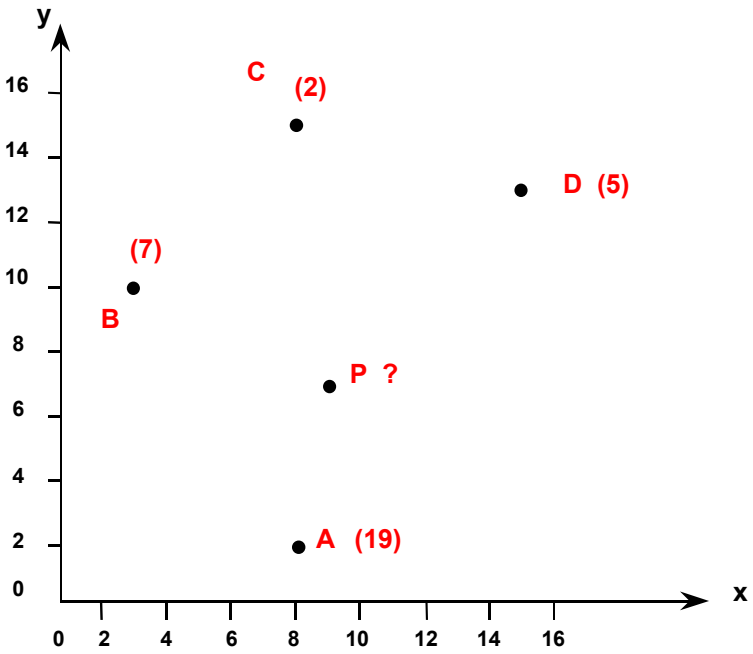
4.2 Formulating NLP

NLPs are formulated similar to LPs and IPs. Decision variables are defined, objective function is formulated and constraints are stated. Notice that in some NLPs there is no constraint at all. Most common applications of NLP are location problems, portfolio management, regression... NLP is very general and very hard to solve

Example 4.3. Unconstrained Facility Location

This is the warehouse location problem with a single warehouse that can be located anywhere in the plane.

The location of the points A, B, C, and D are (8,2), (3,10), (8,15), and (14,13), respectively. The daily demands of these points are 19, 7, 2, and 5. Costs are proportional to the distance. Where should be the warehouse?



Answer

Define (x,y) as the coordinates of the warehouse.

The objective is to minimize the total weighted distance of the points to the warehouse.

Distances can be calculated as $d(P,A) = [(x - 8)^2 + (y - 2)^2]^{1/2}$; $d(P,B) = [(x - 3)^2 + (y - 10)^2]^{1/2}$; $d(P,C) = [(x - 8)^2 + (y - 15)^2]^{1/2}$; $d(P,D) = [(x - 14)^2 + (y - 13)^2]^{1/2}$

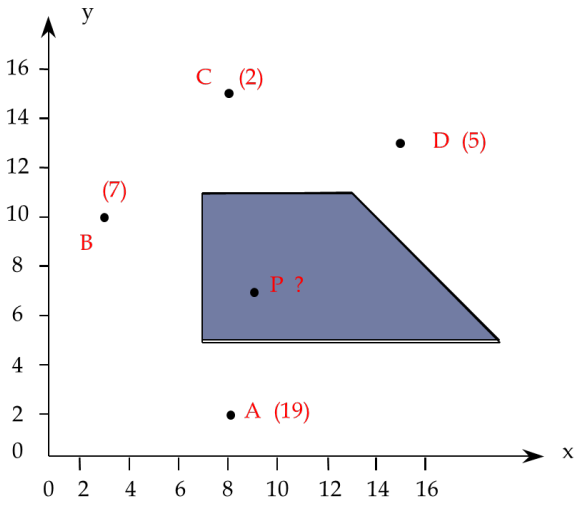
Then NLP would be

minimize $19[(x - 8)^2 + (y - 2)^2]^{1/2} + 7[(x - 3)^2 + (y - 10)^2]^{1/2} + 2[(x - 8)^2 + (y - 15)^2]^{1/2} + 5[(x - 14)^2 + (y - 13)^2]^{1/2}$

subject to: P is unconstrained

Example 4.4. Constrained Facility Location

What happens if P must be within a specified region?



Answer

minimize $19[(x - 8)^2 + (y - 2)^2]^{1/2} + 7[(x - 3)^2 + (y - 10)^2]^{1/2} + 2[(x - 8)^2 + (y - 15)^2]^{1/2} + 5[(x - 14)^2 + (y - 13)^2]^{1/2}$

subject to: $x \geq 7$

$5 \leq y \leq 11$

$x + y \leq 24$

Example 4.5. Tire Production **(Winston 11.2, p. 624)**

Firerock produces rubber used for tires by combining three ingredients: rubber, oil, and carbon black. The cost in cents per pound of each ingredient is 4, 1, and 7, respectively.

The rubber used in automobile tires must have a hardness of between 25 and 35, an elasticity of at 16, a tensile strength of at least 12.

To manufacture a set of four automobile tires, 100 pounds of product is needed. The rubber to make a set of tires must contain between 25 and 60 pounds of rubber and at least 50 pounds of carbon black.

If we define:

R as pounds of rubber in mixture used to produce four tires,

O as pounds of oil in mixture used to produce four tires; and

C as pounds of carbon black used to produce four tires;

then the statistical analysis has shown that the hardness, elasticity, and tensile strength of a 100-pound mixture of rubber, oil, and carbon black is:

Tensile strength = $12.5 - .10O - .001O^2$

Elasticity = $17 + .35R - .04O - .002R^2$

Hardness = $34 + .1R + .06O - .3C + .001RO + .005O^2 + .001C^2$

Formulate an NLP whose solution will tell Firerock how to minimize the cost of producing the rubber product needed to manufacture a set of automobile tires.

Answer

After defining TS (Tensile Strength), E (Elasticity), H (Hardness of mixture), NLP would be:

min $4R + O + 7C$

s.t. $TS = 12.5 - .10O - .001O^2$

$E = 17 + .35R - .04O - .002R^2$

$H = 34 + .1R + .06O - .3C + .001RO + .005O^2 + .001C^2$

$R + O + C = 100$

$25 < R < 60$

$O > 0$

$C > 50$

$TS > 12$

$E > 16$

$25 < H < 35$

Example 4.6. Engineering Design *(Rardin 14.1, p. 794)*

A closed cylindrical tank is being designed to carry at least 20 cubic feet of chemicals. Metal for the top and sides costs $2 per square foot, but the heavier metal of the base costs $8 per square foot. Also, the height of the tank can be no more than twice its diameter to keep it from being top heavy. Formulate an NLP to a design of minimum cost.

Answer

The decision variables:

d: diameter of the tank

h: height of the tank

NLP:

$$\min \quad 2(\pi dh+\pi d^2/4) + 8(\pi d^2/4) \qquad \text{[metal cost]}$$

$$\text{s.t.} \quad \pi h d^2/4 \geq 20 \qquad \text{[volume]}$$

$$h \leq 2d \qquad \text{[height to diameter ratio]}$$

$$h, d \geq 0$$

0-1 IPs as NLPs

$$\text{minimize} \quad \sum_j c_j x_j$$

$$\text{subject to} \quad \sum_j a_{ij} x_j = b_i \quad \text{for all } i$$

$$x_j \text{ is 0 or 1 for all } j$$

is "nearly" equivalent to

$$\text{minimize} \quad \sum_j c_j x_j + 10^6 \sum_j x_j (1 - x_j).$$

$$\text{subject to} \quad \sum_j a_{ij} x_j = b_i \quad \text{for all i}$$

$$0 \leq x_j \leq 1 \text{ for all j}$$

4.3 Review of Differential Calculus

The idea of ***limit***:

The equation

$$\lim_{x \to a} f(x) = c$$

means that as x gets closer to a (but not equal to a), the value of $f(x)$ gets arbitrarily close to c.

A function $f(x)$ is ***continuous*** at a point if

$$\lim_{x \to a} f(x) = f(a)$$

If $f(x)$ is not continuous at $x=a$, we say that $f(x)$ is ***discontinuous*** (or has a discontinuity) at a.

The ***derivative*** of a function $f(x)$ at $x = a$ [written $f'(a)$] is defined to be

$$\lim_{\Delta x \to 0} \frac{f(a+\Delta x) - f(a)}{\Delta x}$$

We may think of $f'(a)$ as the slope of $f(x)$ at $x = a$

If we begin at $x = a$ and increase x by a small amount Δ (Δ may be positive or negative), then $f(x)$ will increase by an amount approximately equal to $\Delta f'(a)$.

- If $f'(a)>0$ then $f(x)$ is increasing at $x = a$
- If $f'(a)<0$ then $f(x)$ is decreasing at $x = a$

We define $f^{(2)}(a) = f''(a)$ to be the derivative of function $f'(x)$ at $x = a$ (if it exists)

Similarly we can define ***higher derivative*** $f^{(n)}(a)$ to be the derivative of $f^{(n-1)}(x)$ at $x = a$.

In the Taylor series expansion of a function $f(x)$, given that $f^{(n+1)}(x)$ exists for every point on the interval $[a,b]$, we can write ***n-th order Taylor series expansion of f(x) about a*** for any h satisfying $0 \le h \le b-a$:

$$f(a+h) = f(a) + \sum_{i=1}^{i=n} \frac{f^{(i)}(a)}{i!} h^i + \frac{f^{(n+1)}(p)}{(n+1)!} h^{n+1}$$

This equation holds will hold for some number p between a and $a+h$

The ***partial derivative*** of $f(x_1, x_2, \ldots x_n)$ with respect to the variable x_i is written as

$$\frac{\partial f}{\partial x_i} = \lim_{\Delta x_i \to 0} \frac{f(x_1, \ldots, x_i + \Delta x_i, \ldots, x_n) - f(x_1, \ldots, x_i, \ldots x_n)}{\Delta x_i}$$

We will also use ***second-order partial derivatives*** and use the notation

$$\frac{\partial^2 f}{\partial x_i \partial x_j}$$

To find second-order partial derivative, we first find partial derivative of $f(x_1, x_2, \ldots x_n)$ with respect to the variable x_i, and then take its partial derivative with respect to x_j.

If the second-order partials exist and are everywhere continuous, then

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}$$

4.4 Convexity and Extreme Points

A set S is a convex set, if for every two points x and y in S, and for every real number $\lambda \in [0,1]$,

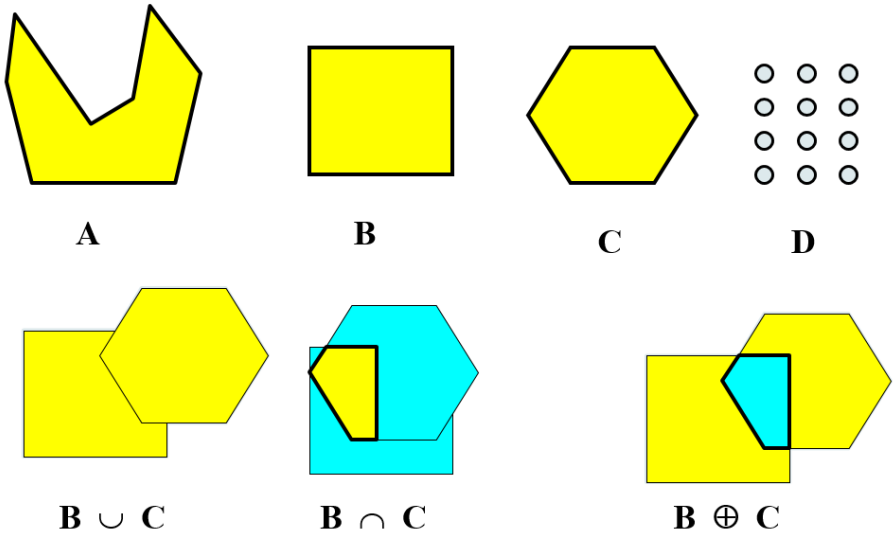
$\lambda x + (1 - \lambda) y$ is in S.

For all x and y, the entire line segment is always in S → Set is convex

We say that an element $v \in S$ is an extreme point (vertex), if v is not the midpoint of any line segment contained in S.

If all constraints are linear, then the feasible region is convex (Hence, the feasible region of an LP is convex).

Example 4.7. Which are convex set?



4.4.1 Convex and Concave Functions

A function $f(x_1, x_2, \ldots, x_n)$ is a ***convex*** function on a convex set S if for any $x' \in S$ and $x'' \in S$

$$f[\lambda x' + (1-\lambda)x''] \leq \lambda f(x') + (1-\lambda) f(x'')$$

holds for $0 \leq \lambda \leq 1$.

A function $f(x_1, x_2, \ldots, x_n)$ is a ***concave*** function on a convex set S if for any $x' \in S$ and $x'' \in S$

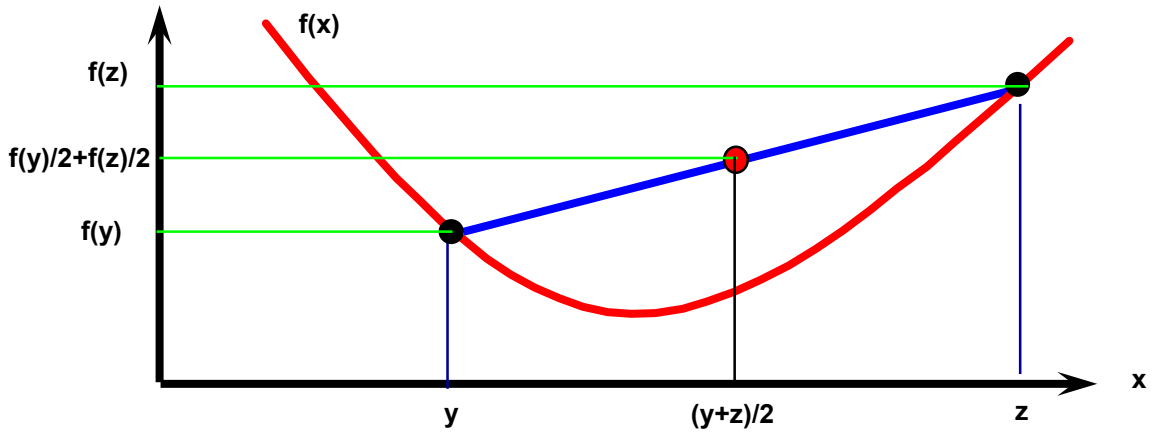
$$f[\lambda x' + (1-\lambda)x''] \geq \lambda f(x') + (1-\lambda) f(x'')$$

holds for $0 \leq \lambda \leq 1$.

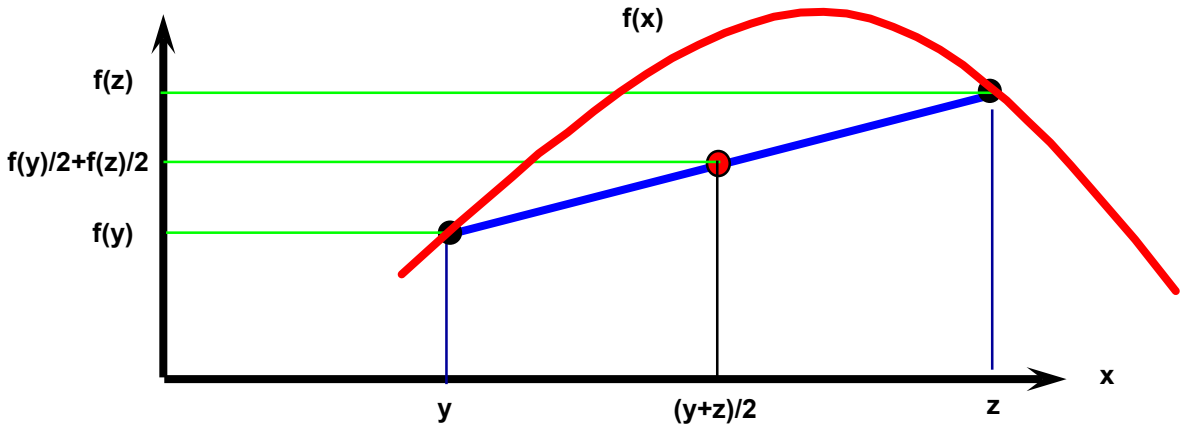
We say ***strict convexity*** if sign is "<"and ***strict concavity*** if sign is ">".

SINGLE VARIABLE CASE

Line joining any two points of the convex function is above the curve.



Line joining any two points of the concave function is below the curve.



Suppose $f''(x)$ exists for all x in a convex set S. Then $f(x)$ is ***convex*** function on S if and only if $f'(x) \geq 0$ for all x in S. We say ***strict convexity*** if sign is ">".

Suppose $f''(x)$ exists for all x in a convex set S. Then $f(x)$ is ***concave*** function on S if and only if $f'(x) \leq 0$ for all x in S. We say ***strict concavity*** if sign is "<".

Example 4.8. Convexity/Concavity

- $f(x) = x^2$ → $f'(x) = 2 > 0$
- $f(x) = -\ln(x)$ for $x > 0$ → $f'(x) = 1/x^2$

NECESSARY DEFINITIONS FOR *n*-VARIABLE CASE

The ***Hessian*** of $f(x_1, x_2, \ldots, x_n)$ is the n x n matrix whose ij^{th} entry is $\dfrac{\partial^2 f}{\partial x_i \partial x_j}$.

An ***i*th *principal minor*** of an $n \times n$ matrix is the determinant of any $i \times i$ matrix obtained by deleting $n - i$ rows and the corresponding $n - i$ columns of the matrix.

The ***k*th *leading principal minor*** of an n x n matrix is the determinant of the k x k matrix obtained by deleting the last $n - k$ rows and columns of the matrix.

We let $\mathbf{H}_k(x_1,x_2...,x_n)$ be the kth leading principal minor of the Hessian matrix evaluated at the point $(x_1,x_2...,x_n)$

A symmetric $n \times n$ matrix is ***positive definite*** if the determinants of all its principal minors are positive (>0).

A symmetric $n \times n$ matrix is ***positive semidefinite*** if the determinants of all its principal minors are nonnegative (≥0).

A symmetric $n \times n$ matrix is ***negative definite*** if the determinants of the principal minors are nonzero and alternating in sign with the first negative.

A symmetric $n \times n$ matrix is ***negative semidefinite*** if the determinants of the principal minors are alternating in sign with the first nonpositive (allows zeros).

Example 4.9. Finding Hessian Matrix and principal minors

If $f(x_1,x_2) = x_1^3 + 2x_1x_2 + x_2^2$, then the value of the Hessian at (x_1,x_2):

$$\mathbf{H}(x_1,x_2) = \begin{bmatrix} 6x_1 & 2 \\ 2 & 2 \end{bmatrix}$$

Principal minors of H:

The 1st principal minors are $6x_1$, and 2

The 2nd principal minor is = $6x_1(2) - 2(2) = 12x_1 - 4$

Example 4.10. Finding principal minors

For the matrix

$$\begin{bmatrix} -2 & -1 \\ -1 & -4 \end{bmatrix}$$

The 1st principal minors are –2 and –4

The 2nd principal minor is (–2)(–4)–(–1)(–1) = 7.

For the matrix $\begin{bmatrix} -2 & 1 & 3 \\ 5 & -4 & 2 \\ 6 & 3 & 7 \end{bmatrix}$

The 1st principal minors are –2, –4, 7

The 2nd principal minors are $\det\left(\begin{bmatrix} -2 & 1 \\ 5 & -4 \end{bmatrix}\right)$, $\det\left(\begin{bmatrix} -2 & 3 \\ 6 & 7 \end{bmatrix}\right)$, $\det\left(\begin{bmatrix} -4 & 2 \\ 3 & 7 \end{bmatrix}\right)$

The 3rd principal minor is $\det\left(\begin{bmatrix} -2 & 1 & 3 \\ 5 & -4 & 2 \\ 6 & 3 & 7 \end{bmatrix}\right)$.

Conditions for Convexity

Suppose $f(x_1, x_2,\ldots, x_n)$ has continuous second-order partial derivatives for each point $x=(x_1, x_2,\ldots, x_n)\in S=R^n$.

Then $f(x_1, x_2,\ldots, x_n)$ is a ***convex*** function on S if and only if for each $x\in S$, ***all principal minors*** of Hessian matrix are ***nonnegative***.

$$f_{11}\geq 0,\quad \begin{vmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{vmatrix} \geq 0,\ \ldots,\ \begin{vmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \cdots & \cdots & & \cdots \\ f_{n1} & f_{n2} & \cdots & f_{nn} \end{vmatrix} \geq 0$$

Suppose $f(x_1, x_2,\ldots, x_n)$ has continuous second-order partial derivatives for each point.

Then $f(x_1, x_2,\ldots, x_n)$ is a ***strict convex*** function on S if and only if for each $x \in S$, ***all principal minors*** of Hessian matrix are ***positive***.

$$f_{11}>0,\ \begin{vmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{vmatrix} >0,\ \ldots,\ \begin{vmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \cdots & \cdots & & \cdots \\ f_{n1} & f_{n2} & \cdots & f_{nn} \end{vmatrix} >0$$

Conditions for Concavity

Suppose $f(x_1, x_2,\ldots, x_n)$ has continuous second-order partial derivatives for each point.

Then $f(x_1, x_2,\ldots, x_n)$ is a ***concave*** function on S if and only if for each $x \in S$ and k=1, 2,…n, ***all nonzero principal minors*** have the ***same sign as*** $(-1)^k$.

$$f_{11}\leq 0,\ \begin{vmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{vmatrix} \geq 0,\ \ldots.,\ \begin{vmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \cdots & \cdots & & \cdots \\ f_{n1} & f_{n2} & \cdots & f_{nn} \end{vmatrix} \leq 0 \ (n \text{ odd})$$

Suppose $f(x_1, x_2,\ldots, x_n)$ has continuous second-order partial derivatives for each point.

Then $f(x_1, x_2,\ldots, x_n)$ is a ***strict concave*** function on S if and only if for each $x \in S$ and k=1, 2,…n, ***all principal minors*** have the ***same sign as*** $(-1)^k$.

$$f_{11}<0,\ \begin{vmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{vmatrix} >0,\ \ldots,\ \begin{vmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \cdots & \cdots & & \cdots \\ f_{n1} & f_{n2} & \cdots & f_{nn} \end{vmatrix} <0 \ (n \text{ odd})$$

Example 4.11. Using the Hessian to Ascertain Convexity or Concavity

Show that $f(x_1,x_2) = x_1^2 + 2x_1x_2 + x_2^2$ is a convex function on $S = R^2$.

Answer

$$H(x_1,x_2) = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$$

The first principal minors of **H** are 2, and 2 > 0.

The second principal minor is (2)(2)–(2)(2) = 0

All principal minors of **H** are nonnegative so $f(x_1,x_2)$ is a convex function on R^2.

Example 4.12. Using the Hessian

Is $f(x_1, x_2, x_3) = x_1^2 + x_2^2 + 2x_3^2 - x_1x_2 - x_1x_3 - x_2x_3$ a convex function on $S = R^3$?

Answer

$$H(x_1,x_2,x_3) = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 4 \end{bmatrix}$$

The *first* principal minors of **H** are 2, 2, and 4 > 0.

The second principal minors:

By deleting row 1 and column 1 of **H**: $det\begin{bmatrix} 2 & -1 \\ -1 & 4 \end{bmatrix} = 7 > 0.$

By deleting row 2 and column 2 of **H**: $det\begin{bmatrix} 2 & -1 \\ -1 & 4 \end{bmatrix} = 7 > 0.$

By deleting row 3 and column 3 of **H**: $det\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} = 3 > 0.$

The *third* principal minor is simply the determinant of **H** itself:

2[(2)(4)-(-1)(-1)] -(-1)[(-1)(4) -(-1)(-1)] +(-1)[(-1)(-1)-(-1)(2)] = 14 - 5 - 3 = 6 > 0

As all principal minors of **H** are positive, f(x1,x2,x3) is a strict convex function.

Example 4.13.

Determine the given function is convex, concave, or neither?

$$f(x_1, x_2, x_3) = -x_1^2 - x_2^2 - 2x_3^2 + 0.5x_1x_2, \;\; S = R^3$$

Example 4.14.

Show that for $S = R^2$, $f(x_1, x_2\,) = x_1^2 + 2x_2^2 - 3x_1x_2$ is not a convex or a concave function.

4.4.2 Local Optima & Saddle Points

The *stationary points* are solutions where all first partial derivatives equal zero (The gradient of $f(\mathbf{x})$: $\nabla f(\mathbf{x})=0$).

If a function *f* has continuous second partial derivatives on a neighborhood of a critical point *a;* then at *a*, *f* has:

A *local maximum* if H(*a*) is *negative definite*

A *local minimum* if H(*a*) is *positive definite*

A *saddle point* is a stationary point that is neither a local maximum nor a local minimum.

Example 4.15. Verifying Saddle Points

Verify that function $f(x_1,x_2)=x_1^2-2x_1-x_2^2$ has a saddle point at **x** = (1, 0)

Answer

$$\frac{\partial f}{\partial x_1} = 2x_1 - 2 = 0, \quad \frac{\partial f}{\partial x_2} = -2x_2 = 0$$

Computing the Hessian gives

$$\mathbf{H}(1,0) = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

The principal minors are 2 and –4 → **H** is neither positive semidefinite nor negative semidefinite.

Thus, **x** violates requirements for both a local maximum and a local minimum → it is a saddle point

4.4.3 The Importance of Convex and Concave Function

Suppose the ***feasible region*** *S* for NLP is a ***convex set***. If the ***objective function*** is ***concave*** on *S*, then any ***local maximum*** for the NLP is an ***optimal solution*** (***global maximum***) to the NLP

Strict concavity implies that the global maximum is ***unique***.

A ***local minimum*** of a ***convex objective function*** on a ***convex feasible region*** is an ***optimal solution*** (***global minimum***)

Strict convexity implies that the global minimum is ***unique***.

4.5 Solving NLPs with One Variable

A NLP with one variable can be formulated as follows:

$$\max(or\ \min) f(x)$$
$$\text{s.t.} \qquad x \in [a,b]$$

To find the optimal solution the local extremums can be analyzed or numerical analysis methods such as Golden Section Search can be used.

4.5.1 Analyzing Local Extremums

All the local maxima (or minima) are found to find the optimal solution. A point that is a local maximum or a local minimum for the NLP is called a local extremum. The optimal solution is the local maximum (or minimum) having the largest (or smallest) value of *f(x)*.

There are three types of points for which the NLP can have a local maximum or minimum (these points are often called ***extremum candidates***).

- CASE 1: Points where a < x < b, $f'(x) = 0$ [called a stationary point of $f(x)$].
- CASE 2: Points where $f'(x)$ does not exist
- CASE 3: Endpoints a and b of the interval $[a,b]$

Example 4.16. Analyzing local extremums to solve one-variable NLP

Let

$$f(x) = 2 - (x-1)^2 \quad for \quad 0 \le x < 3$$
$$f(x) = -3 + (x-4)^2 \quad for \quad 3 \le x \le 6$$

Find

$$\max f(x)$$
$$\text{s.t. } 0 \le x \le 6$$

4.5.2 Golden Section Search

It may be that $f'(x)$ does not exist, or it may be difficult to solve the equation $f'(x) = 0$.
In this case, the ***Golden Section Method*** (***GSS***) can be used to solve NLP if the function is a ***unimodal*** function.

Unimodal Function

*A single variable function f is **unimodal** if there is at most one local maximum (or at most one local minimum)*

Unimodality is a weaker requirement than convexity or concavity.

A unimodal objective function need not be either convex or concave

Both strict convex objective function in minimize problems and strict concave objective function in maximize problems are unimodal

---

A function *f(x)* is unimodal on $[a,b]$ if for some point x' on $[a,b]$, *f(x)* is strictly increasing on $[a,x']$ and strictly decreasing on $[x',b]$ (*maximization problem*): x' denote the optimal solution to NLP.
The optimal solution of the NLP is some point on the interval $[a,b]$.
By evaluating *f(x)* at two points x_1 and x_2 on $[a,b]$ (assume $x_1<x_2$), we may reduce the size of the interval in which the solution to the NLP must lie.
After evaluating $f(x_1)$ and $f(x_2)$, it can be shown that the optimal solution will lie in a subset of $[a,b]$.

Case 1: $f(x_1) < f(x_2)$ and $x' \in (x_1,b]$

Case 2: $f(x_1) = f(x_2)$ and $x' \in [a,x_2]$

Case 3: $f(x_1) > f(x_2)$ and $x' \in [a,x_2)$

The interval in which x' must lie – either $[a,x_2)$ or $(x_1, b]$ - is called the ***interval of uncertainty***.
Many search algorithms use these ideas to reduce the interval of uncertainty.
Most of these algorithms proceed as follows:

1. Begin with the region of uncertainty for x being $[a,b]$. Evaluate $f(x)$ at two judiciously chosen points x_1 and x_2.
2. Determine which of Cases 1-3 holds, and find a reduced interval of uncertainty.
3. Evaluate $f(x)$ at two new points (the algorithm specifies how the two new points are chosen). Return to step 2 unless the length of the interval of uncertainty is sufficiently small.

GSS begins reducing the interval of uncertainty by evaluating $f(x)$ at two points x_1 and x_2

$x_1 = b - r(b - a)$

$x_2 = a + r(b - a)$

where $r = 0.618$ [unique positive root of $r^2+r=1$]

Then GSS generates two new points with the following moves

- New LH point: Move a distance equal to a fraction r of the current interval of uncertainty from the right endpoint of the interval of uncertainty.
- New RH point: Move a distance equal to a fraction r of the current interval of uncertainty from the left endpoint of the interval of uncertainty.

Example 4.17. GSS *(Winston 11.5, p. 652)*

Use GSS to find

$\max -x^2 - 1$

s.t. $-1 \le x \le 0.75$

with the final interval of uncertainty having a length less than 0.25.

Answer

$a = -1$, $b = 0.75$, and $b - a = 1.75$

To determine the number k of iterations of GSS that must be performed, we solve for k using $1.75(0.618^k)<0.25$.

Taking logarithms to base e of both sides: $k>4.06$

Thus, five iterations of GSS must be performed

Please refer to Winston for iterations

Optimal solution must lie within the interval (–0.0762,0.0815]

(*The actual maximum occurs for x = 0*)

4.6 Solving Unconstrained NLP

In an unconstraint NLP, it is the objective function to be analyzed. For this, analytic approaches as well as numerical analysis methods can be used.

4.6.1 Analyzing Local Extremums

First Order Necessary Condition

Every unconstrained local optimum of a smooth (differentiable) objective function must be a ***stationary*** point.

$$\frac{\partial f(\bar{x})}{\partial x_i} = 0$$

Second Order Necessary Condition

The Hessian matrix of a smooth function *f* is a ***negative semidefinite*** at every unconstrained local maximum.

The Hessian matrix is a ***positive semidefinite*** at every unconstrained local minimum.

Sufficient Condition for Local Optima

A stationary point of a smooth function *f* is an unconstrained local maximum if the Hessian matrix is ***negative definite***.

A stationary point is an unconstrained local minimum if the Hessian matrix is ***positive definite***.

Sufficient Condition for Global Optima

If *f* is a ***convex*** function, every unconstrained local minimum of *f* is an unconstrained global minimum.

If *f* is ***concave***, every unconstrained local maximum is an unconstrained global maximum.

Example 4.18. Verifying Global Optima

Find the unconstrained global optimum for $f(x) = 20 - x^2 + 6x$

Answer

Differentiating

$f'(x) = -2x + 6 = 0$ → $x = 3$ is a stationary point

$f''(x) = -2 < 0$ → $f(x)$ is strictly concave

Thus, $x = 3$ is the unconstrained global maximum.

Example 4.19.

Find all local maxima, local minima, and saddle points for $f(x, y) = x^2 - x^2y + (y + 1)^2$

Answer

Find points where $\frac{\partial f(x,y)}{\partial x} = 0$ and $\frac{\partial f(x,y)}{\partial y} = 0$.

(x, y) = (0, -1) → local min

(x, y) = (2, 1) → not local extremum

(x, y) = (-2, 1) → not local extremum

Example 4.20. Monopolistic Pricing *(Winston 11.6., p. 656)*

A monopolist producing a single product has two types of customers. If q_1 units are produced for customer 1, then customer 1 is willing to pay a price of $70 - 4q_1$ dollars. If q_2 units are produced for customer 2, then customer 2 is willing to pay a price of $150 - 15q_2$ dollars. For $q > 0$, the cost of manufacturing q units is $100 + 15q$ dollars. To maximize profit, how much should the monopolist sell to each customer?

4.6.2 Gradient Search

Consider a function $f(x_1, x_2,\ldots x_n)$, all of whose partial derivatives exist at every point.

A **gradient vector** for $f(x_1, x_2,\ldots x_n)$, written $\nabla f(\mathbf{x})$, is given by

$$\nabla f(x)=\left[\frac{\partial f(x)}{\partial x_1},\frac{\partial f(x)}{\partial x_2},\cdots,\frac{\partial f(x)}{\partial x_n}\right]$$

In many problems, it may be difficult to find a stationary point where all first partial derivatives equal zero (The gradient of $f(\mathbf{x})$: $\nabla f(\mathbf{x})=\mathbf{0}$).

Gradient search can be used to approximate a function's stationary point by using the following principle:

- When objective function gradient $\nabla f(\mathbf{x})\neq\mathbf{0}$,

  $\Delta\mathbf{x}= \nabla f(\mathbf{x})$ is an improving direction for a maximize objective f, and

  $\Delta\mathbf{x}= -\nabla f(\mathbf{x})$ is an improving direction for a minimizing f.

First order Taylor series computations of expression guarantee improvement with sufficiently small steps in direction $\Delta\mathbf{x}$.

Gradient search is sometimes called the ***method of steepest ascent*** for maximize (***steepest descent*** for minimize) problems.

Although gradient search may produce good initial progress, zigzagging as it approaches a stationary point makes the method too slow and unreliable to provide satisfactory results in many unconstrained NLP applications.

The Method of Steepest Ascent (Descent)

Given a vector $\mathbf{x} = (x_1, x_2,\ldots, x_n) \in R_n$, the ***length*** of $\mathbf{x}$ (written $||x||$) is $(x_1^2+x_2^2+\ldots+x_n^2)^{1/2}$. For any vector $\mathbf{x}$, the unit vector $\mathbf{x}/||\mathbf{x}||$ is called the ***normalized*** version of $\mathbf{x}$.

Suppose we are at a point $\mathbf{v}$ and we move from $\mathbf{v}$ a small distance δ in a direction $\mathbf{d}$.

Then for a given δ, the maximal increase in the value of $f(\mathbf{x})$ will occur if $\mathbf{d}=\nabla f(\mathbf{x})/||\nabla f(\mathbf{x})||$

If we move a small distance away from **v** and we want $f(\mathbf{x})$ to increase as quickly as possible, then we should move in the direction of $\nabla f(\mathbf{v})$.

STEPS

1. Begin at any point $\mathbf{v}_0$.
2. The maximum possible improvement in the value of f (for a max problem) that can be attained by moving away from $\mathbf{v}_0$ in the direction of $\nabla f(\mathbf{v}_0)$ results from moving to $\mathbf{v}_1=\mathbf{v}_0+t_0\nabla f(\mathbf{v}_0)$ where t_0 solves the following one variable optimization problem

   max $f(\mathbf{v}_0+t_0\nabla f(\mathbf{v}_0))$

   s.t. $t_0 \geq 0$
3. If $||\nabla f(\mathbf{v}_1)||$ is small, terminate the algorithm ($\mathbf{v}_1$ is near a stationary point).
4. If $||\nabla f(\mathbf{v}_1)||$ is not sufficiently small (i.e. not less than 0.01), then move away from $\mathbf{v}_1$ a distance t_1 in the direction of $||\nabla f(\mathbf{v}_1)||$.
5. Continue in this fashion until reaching a point $\mathbf{v}_a$ having $||\nabla f(\mathbf{v}_a)||$ sufficiently small.

Example 4.21. Steepest Ascent *(Winston 11.7., p. 663)*

Use the method of steepest ascent to approximate the solution to

$$Max\ z = -(x_1-3)^2-(x_2-2)^2 = f(x_1,x_2)$$
$$s.t.\ (x_1,x_2) \in R^2$$

Consider $\mathbf{v}_0$ = (1, 1).

4.6.3 Newton's Method

Newton's method, also known as the Newton–Raphson method, uses Newton step to find a new point. To improve on the slow zigzagging progress characteristics of gradient search, an obvious possibility is extending to the second order Taylor approximation → Newton Step

Newton steps $\Delta\mathbf{x}$, which move to a stationary point (if there is one), of the second order Taylor series approximation to $f(\mathbf{x})$ at the current point $\mathbf{x}^{(t)}$ are obtained by solving the linear equation system:

$$\mathbf{H}(\mathbf{x}^{(t)})\,\Delta\mathbf{x} = -\nabla f(\mathbf{x}^{(t)})$$

If Newton's method converges to a local optimum, it usually does so in many fewer steps than first order procedures such as gradient search.

Computing both first and second partial derivatives plus solving a linear system of equations at each iteration makes Newton's method computationally burdensome as the dimension of the decision vector becomes large.

Notice that if there is single variable, Δx can be found as $\Delta x = \frac{f'(x)}{f''(x)}$

STEPS

Newton's method proceeds by repeating the process:

- Uses 1st and 2nd partial derivatives at the current point to compute a Newton step
- Updates the solution with that step:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x}$$

Algorithm stops if gradient norm at that point is less than stopping tolerance

$$||\nabla f(\mathbf{x}^{(t)})|| < \in_t$$

Example 4.22. Newton's method

Solve the following NLP using Newton's method starting from the point $x^0 = (0,1)$

$$Min\, f(x,y) = (x+1)^4 + xy + (y+1)^4$$

4.7 Solving Constrained NLP

There are various methods in the literature and in practice to solve Constrained NLPs such as

- Lagrange Multiplier Methods
- KKT Optimality Conditions
- Penalty and Barrier Methods
- Reduced Gradient Algorithms
- Quadratic Programming Methods
- Separable Programming Methods

We will cover fundamentals ones in the course.

4.7.1 Lagrange Multipliers

Lagrange multiplier solution techniques are most easily applied to models in equality constrained format:

min or max $f(\mathbf{x})$

s.t. $g_i(\mathbf{x}) = b_i$ for all i=1, 2, ..., m

These techniques address NLPs in pure equality form.

That is they consider only a set of constraints assumed active, which can be taken as equalities.

Lagrangian Function

The Lagrangian function associated with an NLP over equality constraints $g_i(\mathbf{x}) = b_i$ is

$$L(\mathbf{x}, \mathbf{v}) = f(\mathbf{x}) + \Sigma_i\, \lambda_i\, [b_i - g_i(\mathbf{x})]$$

where λ_i is the Lagrange multiplier for constraint i.

Lagrangian Stationary Points

Points where gradient $\nabla L(\mathbf{x}^*, \lambda^*) = 0$ are stationary points of the Lagrangian. Therefore solution $(\mathbf{x}^*, \lambda^*)$ is a stationary point of Lagrangian function $L(\mathbf{x}, \lambda)$ if it satisfies

$g_i(\mathbf{x}^*) = b_i$ for all i

$\sum_i \nabla g_i(\mathbf{x}^*)\, \lambda_i^* = \nabla f(\mathbf{x}^*)$ for all j or $\sum_i \frac{\partial g_i}{\partial x_j} \lambda_i = \frac{\delta f}{\delta x_j}$ for all j.

Optimum Point

If $(\mathbf{x}^*, \lambda^*)$ is a stationary point of the Lagrangian function $L(\mathbf{x}, \lambda)$ and $\mathbf{x}^*$ is an unconstrained optimum of $L(\mathbf{x}, \lambda^*)$, then $\mathbf{x}^*$ is an optimum of the corresponding equality constrained NLP.

Interpretation of Lagrange Multipliers

The optimal Lagrange multiplier, λ_i^* associated with constraint $g_i(\mathbf{x}) = b_i$ can be interpreted as the rate of change in optimal value per unit increase in RHS b_i.

Limitations of the Lagrangian Approach

Stationary point conditions can be solved only if they are linear or very simple nonlinear functions. In other cases, solving these conditions may be more difficult than directly searching for an optimal solution to the original model.

Example 4.23. Lagrange Multiplier Method *(Winston 11.8, p.667)*

A company is planning to spend $10,000 on advertising. It costs $3,000 per minute to advertise on television and $1,000 per minute to advertise on radio. If the firm buys x minutes of television advertising and y minutes of radio advertising, then its revenue in thousands of dollars is given by $f(x,y) = -2x^2 - y^2 + xy + 8x + 3y$. How can the firm maximize its revenue?

4.7.2 KKT Optimality Conditions

Historically, W. ***Karush*** was the first to develop the KKT conditions in 1939 as part of his M.S. thesis. The same conditions were developed independently in 1951 by W. ***Kuhn*** and A. ***Tucker***.

Lagrangian Approach deals only with equality constraints. ***KKT conditions*** address the full differentiable NLP.

Full Differentiable NLPs have the general form:

min or max	$f(\mathbf{x})$	
s.t.	$g_i(\mathbf{x}) \geq b_i$	for all $i \in G$
	$g_i(\mathbf{x}) \leq b_i$	for all $i \in L$
	$g_i(\mathbf{x}) = b_i$	for all $i \in E$

where f and all g_i are differentiable functions.

Complementary Slackness

Either inequality constraints should be active at a local optimum or the corresponding Lagrange variable should be 0:

$$\lambda_i \, [b_i - g_i(\mathbf{x})] = 0 \quad \text{for all inequalities } i$$

Lagrange Multiplier Sign Restrictions

Lagrange multipliers λ_i on constraints i of full differentiable NLP should satisfy the following sign restrictions:

Objective	***i* is ≤**	***i* is ≥**	***i* is =**
Minimize	$\lambda_i \leq 0$	$\lambda_i \geq 0$	urs
Maximize	$\lambda_i \geq 0$	$\lambda_i \leq 0$	urs

KKT Points

Solutions **x** and λ satisfy the ***KKT conditions*** for differentiable NLP if they fulfill

- complementary slackness conditions
- sign restrictions
- gradient equation ($\sum_i \nabla g_i(\mathbf{x}) \, \lambda_i = \nabla f(\mathbf{x})$)
- primal constraints of the original NLP model

Any **x** for which there exist a corresponding λ satisfying these conditions is called a *KKT point*.

Necessity of KKT Conditions for Optimality

A local optimum solution of a constrained differentiable NLP must be a KKT point if

- All constraints are linear or
- The gradients of all constraints active at the local optimum are linearly independent

---

LICQ (Linear Independence Constraint Qualification):

A collection of vectors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k$ is considered LI if $\sum_i \lambda_i \mathbf{x}_i = 0$ implies that $\lambda_i = 0$ for all i's.

Sufficiency of KKT Conditions for Optimality

If **x** is a KKT point of a convex program, **x** is a global optimum.

CONVEX PROGRAM

A constrained NLP is a convex program if

- f is convex for a min. or concave for a max.
- each g_i of a ≥ constraint is concave
- each g_i of a ≤ constraint is convex
- each g_i of a ≥ constraint is linear

Example 4.24. KKT *(Winston 11.9, p.676)*

A monopolist can purchase up to 20 oz of a chemical for \$10/oz. At a cost of \$3/oz, the chemical can be processed into an ounce of product 1; or, at a cost of \$5/oz, the chemical can be processed into an ounce of product 2. If x_1 oz of product 1 are produced, it sells for a price of \$30-x_1 per ounce. If x_2 oz of product 2 are produced, it sells for a price of \$50-2x_2 per ounce. Determine how the monopolist can maximize profits.

Answer

Define x_1 = ounces of product 1 produced, x_2 = ounces of product 2 produced, x_3 = ounces of chemical processed. The following NLP can be used to solve the problem:

$$Max\ z = x_1(30 - x_1) + x_2(50 - 2x_2) - 3x_1 - 5x_2 - 10x_3$$

$$st \quad x_1 + x_2 - x_3 \leq 0$$

$$x_3 \leq 20$$

KKT conditions for the given NLP:

$$30 - 2x_1 - 3 - \lambda_1 = 0$$

$$50 - 4x_2 - 5 - \lambda_1 = 0$$

$$-10 + \lambda_1 - \lambda_2 = 0$$

$$\lambda_1(-x_1 - x_2 + x_3) = 0$$

$$\lambda_2(20 - x_3) = 0$$

$$\lambda_1 \geq 0$$

$$\lambda_2 \geq 0$$

There are four cases to consider:

Case 1: $\lambda_1 = \lambda_2 = 0$

Case 2: $\lambda_1 = 0$, $\lambda_2 > 0$

Case 3: $\lambda_1 > 0$, $\lambda_2 = 0$

Case 4: $\lambda_1 > 0$, $\lambda_2 > 0$

In order to find the optimal solution, we have to analyze all the cases to find out which one satisfy the KKT conditions.

4.8 Solving NLP on a PC

LINGO may be used to solve NLPs. ***Excel Solver*** may also be used to solve NLPs.

LINGO and Solver use calculus-based methods.

For NLPs having multiple local optimal solutions, LINGO and Solver may fail to find the optimal solution as they may pick a local extremum that is not a global extremum.

5. INTRODUCTION TO INTERIOR POINT METHODS

Interior Point Methods (***IPM***) still follow the improving search paradigm for LP, but they employ moves quite different from those in simplex method.

Much more effort turns out to be required per move (iteration) with IPM but the number of moves decreases dramatically.

The simplex algorithm is an ***exponential time algorithm*** for solving LPs. If an LP of size n is solved by the simplex, then there exists a positive number a such that for any n, the simplex algorithm will find the optimal solution in a time of at most $a2^n$.

IPM, on the other hand, is a **polynomial time algorithm**. This implies that if an LP of size n is solved by an IPM, then there exist positive numbers b and c such that for any n, LP can be solved in a time of at most bn^c

Instead of staying on the boundary of the feasible region and passing from extreme point to extreme point, IPM proceed directly across the interior.

IPM begin at and move through a sequence of interior feasible solutions, converging to the boundary of the feasible region only at an optimal solution.

Popular methods

- Karmarkar's Projective Transformation
- Affine Scaling
- Log Barrier

Interior in LP Standard Form

IPM are applied to an LP in the following standard form:

$$\max \quad \mathbf{cx}$$
$$\text{s.t.} \quad \mathbf{Ax} = \mathbf{b}$$
$$\mathbf{x} \geq \mathbf{0}$$

A feasible solution for an LP in standard form is an ***interior point*** if every component (variable) of the solution that can be positive in any feasible solution is strictly positive in the given point.

Projecting to Deal with Equality Constraints

A move direction $\Delta\mathbf{x}$ is feasible for equality constraints $\mathbf{Ax}=\mathbf{b}$ if it satisfies $\mathbf{A}\Delta\mathbf{x}=\mathbf{0}$.

The ***projection*** of a move vector **d** on a given system of equalities is a direction preserving those constraints and minimizing the total squared difference between its components and those of **d**.

The projection of direction **d** onto conditions **AΔx=0** preserving linear inequalities **Ax=b** can be computed as **Δx=Pd** where ***projection matrix*** $\mathbf{P} = \mathbf{I} - \mathbf{A}^T (\mathbf{A}\mathbf{A}^T)^{-1} \mathbf{A}$

Improvement with Projected Directions

For maximization problems, the projection **Δx=Pc** of (nonzero) objective function vector **c** onto equality constraints **Ax=b** is an improving direction at every **x**.

For minimization problems, the projection **Δx = -Pc** of (nonzero) objective function vector **c** onto equality constraints **Ax=b** is an improving direction at every **x**.

Affine Scaling Approach

(x_1', x_2', x_3'): initial solution →

Diagonal matrix **D** = $\begin{bmatrix} x_1' & 0 & 0 \\ 0 & x_2' & 0 \\ 0 & 0 & x_3' \end{bmatrix}$

such that $\mathbf{x} = \mathbf{D}\tilde{\mathbf{x}}$

Rescaled variable: $\tilde{\mathbf{x}} = \mathbf{D}^{-1}\mathbf{x}$ [*Centering scheme (1, 1, 1)*]

For new coordinates

$$\tilde{\mathbf{A}} = \mathbf{A}\mathbf{D}$$
$$\tilde{\mathbf{c}} = \mathbf{D}\mathbf{c}$$

Projection matrix: $\mathbf{P} = \mathbf{I} - \tilde{\mathbf{A}}^T (\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T)^{-1} \tilde{\mathbf{A}}$

Projected gradient: $\mathbf{c}_P = \pm \mathbf{P}\tilde{\mathbf{c}}$

Update → $\tilde{\mathbf{x}}_{new} = \tilde{\mathbf{x}} + \frac{\alpha}{v}\mathbf{c}_P$

where *v* is the absolute value of the negative component of $\mathbf{c}_p$ having largest absolute value and a is arbitrarily chosen as 0.8

α measures the fraction used of the distance that could be moved before the feasible region is left. An α value close to upper bound of 1 is good for giving a relatively large step toward optimality on the current iteration. However, the problem with a value too close to 1 is that the next trial solution then is jammed against a constraint boundary, thereby making it difficult to take large improving steps during subsequent iterations.

In original coordinates: $\mathbf{x}_{new} = \mathbf{D}\,\tilde{\mathbf{x}}_{new}$

When the solution value is no longer changing very much, algorithm stops

Example 5.1. Frannie's Firewood *(Rardin 6.1., p. 274)*

Solve the following LP using Affine Scaling Approach.

max 90 x_1 + 150 x_2

s.t. 0.5 x_1 + x_2 ≤ 3

x_1, x_2 ≥ 0

Standard form:

max 90 x_1 + 150 x_2

s.t 0.5 x_1 + x_2 + s_1 = 3

x_1, x_2 , s_1 ≥ 0

consider $x^{(0)} = (1, 0.5, 2)^T$

Answer to the question will be uploaded to Ninova as an Excel file.

6. DETERMINISTIC DYNAMIC PROGRAMMING

Dynamic Programming (DP) is a technique that can be used to solve many optimization problems. In most applications, DP obtains solutions by working backward from the end of a problem toward the beginning, thus breaking up a large, unwieldy problem into a series of smaller, more tractable problems.

6.1 Two Puzzles

In this section, it is shown how working backward can make a seemingly difficult problem almost trivial to solve.

Example 6.1. Match Puzzle *(Winston 18.1, p.961)*

30 matches on a table
Two players: my opponent and me
Pick up 1, 2, or 3 matches
I begin, my opponent continues, I continue, …
The player who picks up the last match is "loser"
How can I be sure of winning the game?

Answer

If I can ensure that it will be my opponent's turn when 1 match remain, I'll certainly win! Working backward one step, if I can ensure that it will be my opponent's turn when 5 matches remain, I'll win: e.g., if he picks up 2 matches, I will pick up 2 matches.
Similarly, if I can force my opponent to play when 5, 9, 13, …. 29 matches remain, I'm sure of victory.
Therefore, I should pick up 1 match.
We have solved this puzzle by working backward from the end of the problem toward the beginning.

Example 6.2. Milk Puzzle *(Winston 18.1, p.961)*

I have a 9-liter cup and a 4-liter cup. My mother has ordered me to bring home exactly 6 liters of milk. How can I accomplish this goal?

Answer

By starting near the end of the problem, I clearly realize that I can solve this problem if I can somehow get 1 liter of milk into 4-liter cup. Then I can fill 9-liter cup and empty 3 liter from this cup into partially filled 4-liter: I'll be left with 6 liter of milk.

The problem may easily be described as in the table (the initial solution is written last, the final solution is written first)

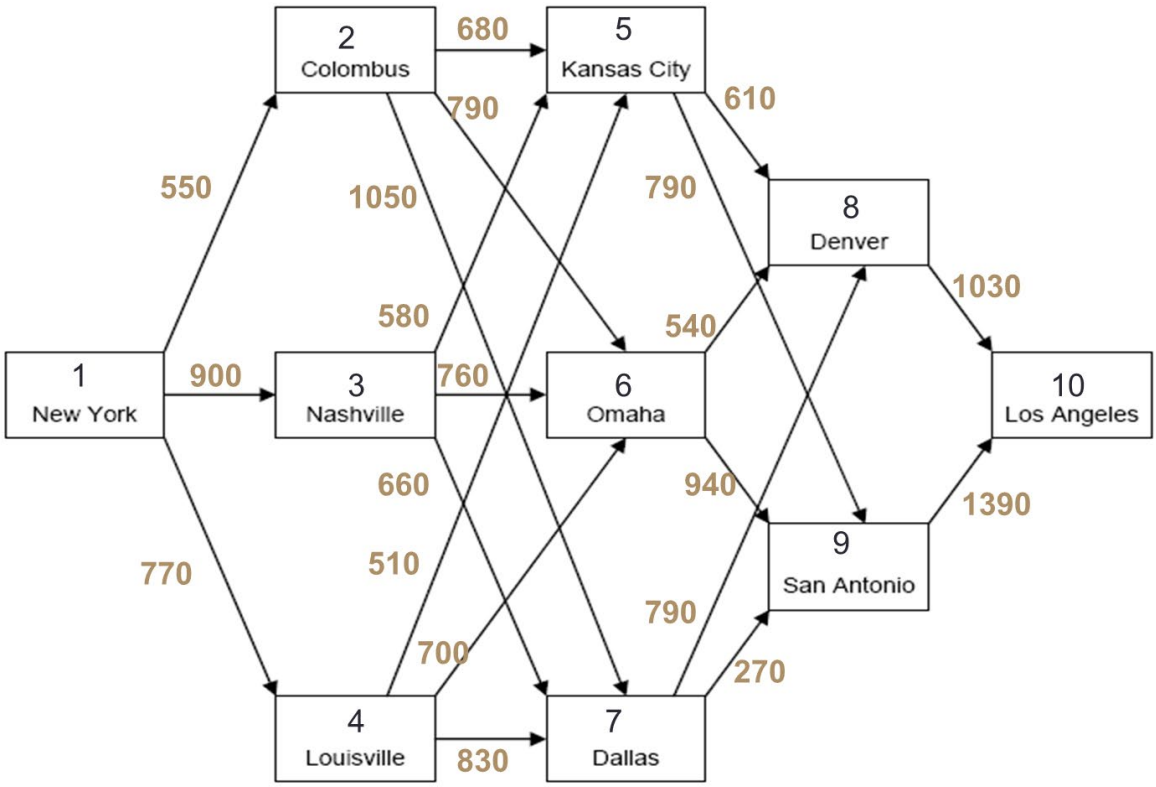
Liters in 9-liter cup	Liters in 4-liter cup
6	0
6	4
9	1
0	1
1	0
1	4
5	0
5	4
9	0
0	0

6.2 A Network Problem

Many applications of DP reduce to finding the shortest (or longest) path that joins two points in a given network. The following example illustrates how DP (working backward) can be used to find the shortest path in a network.

Example 6.3. Shortest Path *(Winston 18.2, p.963)*

Joe Cougar lives in New York City, but he plans to drive to Los Angeles to seek fame and fortune. Joe's funds are limited, so he has decided to spend each night on his trip at a friend's house. Joe has friends in Columbus, Nashville, Louisville, Kansas City, Omaha, Dallas, San Antonio, and Denver. Joe knows that after one day's drive he can reach Columbus, Nashville, or Louisville. After two days of driving, he can reach Kansas City, Omaha, or Dallas. After three days of driving, he can reach San Antonio or Denver. Finally, after four days of driving, he can reach Los Angeles. To minimize the number of miles traveled, where should Joe spend each night of the trip? The actual road mileages between cities are given in the following Figure.



Answer

Joe needs to know the shortest path between New York and Los Angeles in the network. We will find it by working backward. We have classified all the cities that Joe can be in at the beginning of *nth* day of his trip as stage *n* cities. For example, because Joe can only be in San Antonio or Denver at the beginning of the fourth day (day 1 begins when Joe leaves New York), we classify San Antonio and Denver as stage 4 cities. The reason for classifying cities according to stages will become apparent later.

The idea of working backward implies that we should begin by solving an easy problem that will eventually help us to solve a complex problem. Hence, we begin by finding the shortest path to Los Angeles from each city in which there is only one day of driving left (Stage 4 cities). Then we use this information to find the shortest path to Los Angeles from each city for which only two days of driving remain (Stage 3 cities). With this information in hand, we are able to find the shortest path to Los Angeles from each city that is three days distant (Stage 2 cities). Finally, we find the shortest path to Los Angeles from each city (there is only one: New York) that is four days away.

To simplify the exposition, we use the numbers 1, 2,..., 10 given in the figure to label the 10 cities. We also define c_{ij} to be the road mileage between city *i* and city *j.* For example, c_{35}=580 is the road mileage between Nashville (3) and Kansas City (5). We let $f_t(i)$ be the length or the shortest path from city *i* to Los Angeles, given that city *i* is a stage *t* city.

STAGE 4 COMPUTATIONS

We first determine the shortest path to Los Angeles from each stage 4 city. Since there is only one path from each stage 4 city to Los Angeles, we immediately see that $f_4(8)=1030$, the shortest path from Denver to Los Angeles simply being the *only* path from Denver to Los Angeles. Similarly, $f_4(9)=1390$, the shortest (and only) path from San Antonio to Los Angeles.

STAGE 3 COMPUTATIONS

We now work backward one stage (to stage 3 cities) and find the shortest path to Los Angeles from each stage 3 city. For example, to determine $f_3(5)$, we note that the shortest path from city 5 to Los Angeles must be one of the following:

Path 1: Go from city 5 to city 8 and take the shortest path from city 8 to city 10.

Path 2: Go from city 5 to city 9 and take the shortest path from city 9 to city 10.

The length of path 1 may be written as $c_{58}+f_4(8)$, and the length of path 2 may be written as $c_{59}+f_4(9)$. Hence, the shortest distance from city 5 to city 10 may be written as

$$f_3(5)=\min\begin{cases} c_{58}+f_4(8)=610+1030=1640^* \\ c_{59}+f_4(9)=790+1390=2180 \end{cases}$$

[The * indicates the choice of arc that attains the $f_3(5)$]

Thus, we have shown that the shortest path from city 5 to city 10 is the path 5-8-10. Note that to obtain this result, we made use of our knowledge of $f_4(8)$ and $f_4(9)$.

Similarly, to find $f_3(6)$, we note that the shortest path to Los Angeles from city 6 must begin by going to city 8 or city 9. This leads us to the following equation:

$$f_3(6)=\min\begin{cases} c_{68}+f_4(8)=540+1030=1570^* \\ c_{69}+f_4(9)=940+1390=2330 \end{cases}$$

Thus, $f_3(6)=1570$, and the shortest path from city 6 to city 10 is the path 6-8-10, to find $f_3(7)$, we note that

$$f_3(7)=\min\begin{cases} c_{78}+f_4(8)=790+1030=1820 \\ c_{79}+f_4(9)=270+1390=1660^* \end{cases}$$

Therefore, $f_3(7)=1660$, and the shortest path from city 7 to city 10 is the path 7-9-10.

STAGE 2 COMPUTATIONS

Given our knowledge of $f_3(5)$, $f_3(6)$, and $f_3(7)$, it is now easy to work backward one more stage and compute $f_2(2)$, $f_2(3)$, and $f_2(4)$ and thus the shortest paths to Los Angeles from city 2, city 3, and city 4. To illustrate how this is done, we find the shortest path (and its length) from city 2 to city 10. The shortest path from city 2 to city 10 must begin by going from city 2 to city 5, city 6, or city 7. Once the shortest path gets to city 5, city 6, or city 7, then it must follow a shortest path from that city to Los Angeles. This reasoning shows that the shortest path from city 2 to city 10 must be one of the following:

Path 1: Go from city 2 to city 5. Then follow a shortest path from city 5 to city 10. A path of this type has a total length of $c_{25}+f_3(5)$.

Path 2: Go from city 2 to city 6. Then follow a shortest path from city 6 to city 10. A path of this type has a total length of $c_{26}+f_3(6)$.

Path 3: Go from city 2 to city 7. Then follow a shortest path from city 7 to city 10. A path of this type has a total length of $c_{27}+f_3(7)$. We may conclude that

$$f_2(2)=\min\begin{cases} c_{25}+f_3(5)=680+1640=2320^* \\ c_{26}+f_3(6)=790+1570=2360 \\ c_{27}+f_3(7)=1050+1660=2710 \end{cases}$$

Thus, $f_2(2)=2320$, and the shortest path from city 2 to city 10 is to go from city 2 to city 5 and then follow the shortest path from city 5 to city 10 (5-8-10). Similarly,

$$f_2(3)=\min\begin{cases} c_{35}+f_3(5)=580+1640=2220^* \\ c_{36}+f_3(6)=760+1570=2330 \\ c_{37}+f_3(7)=660+1660=2320 \end{cases}$$

Thus, $f_2(3)=2220$, and the shortest path from city 3 to city 10 consists of arc 3-5 and the shortest path from city 5 to city 10 (5-8-10).

In similar fashion,

$$f_2(4)=\min\begin{cases} c_{45}+f_3(5)=510+1640=2150^* \\ c_{46}+f_3(6)=700+1570=2270 \\ c_{47}+f_3(7)=830+1660=2490 \end{cases}$$

Thus, $f_2(4)=2150$, and the shortest path from city 4 to city 10 consists of arc 4-5 and the shortest path from city 5 to city 10 (5-8-10).

STAGE 1 COMPUTATIONS

We can now use our knowledge of $f_2(2)$, $f_2(3)$, and $f_2(4)$ to work backward one more stage to find $f_1(1)$ and the shortest path from city 1 to 10. Note that the shortest path from city 1 to city 10 must be one of the following:

Path 1: Go from city 1 to city 2 and then follow a shortest path from city 2 to city 10. The length of such a path is $c_{12}+f_2(2)$.

Path 2: Go from city 1 to city 3 and then follow a shortest path from city 3 to city 10. The length of such a path is $c_{13}+f_2(3)$.

Path 3: Go from city 1 to city 4 and then follow a shortest path from city 4 to city 10. The length of such a path is $c_{14}+f_2(4)$. It now follows that

$$f_1(1)=\min\begin{cases} c_{12}+f_2(2)=550+2320=2870^* \\ c_{13}+f_2(3)=900+2220=3120 \\ c_{14}+f_2(4)=770+2150=2920 \end{cases}$$

Optimal Solution and Report

Thus, $f_1(1) = 2870$, and the shortest path from city 1 to city 10 goes from city 1 to city 2 and then follows the shortest path from city 2 to city 10. Checking back to the $f_2(2)$ calculations, we see that the shortest path from city 2 to city 10 is 2-5-8-10.

The shortest path from New York to Los Angeles passes through New York, Columbus, Kansas City, Denver, and Los Angeles.

6.3 Characteristics of DP Applications

The characteristics of Network Problem are common to most applications of DP:

1. *The problem can be divided into stages with a decision required at each stage.* In Network Problem, stage *t* consisted of those cities where Joe could be at the beginning of day *t* of his trip. As we will see, in many DP problems, the stage is the amount of time that has elapsed since the beginning of the problem. We note that in some situations, decisions are not required at every stage.
2. *Each stage has a number of states associated with it.* By a **state**, we mean the information that is needed at any stage to make an optimal decision. In Network Problem, the state at stage *t* is simply the city where Joe is at the beginning of day *t*. For example, in stage 3, the possible states are Kansas City, Omaha, and Dallas. Note that to make the correct decision at any stage, Joe doesn't need to know how he got to his current location. For example, if Joe is in Kansas City, then his remaining decisions don't depend on how he goes to Kansas City; his future decisions depend on the fact that he is now in Kansas City.
3. *The decision chosen at any stage describes how the state at the current stage is transformed into the state at the next stage.* In Network Problem, Joe's decision at any stage is simply the next city to visit. This determines the state at the next stage in an obvious fashion. In many problems, however, a decision does not determine the next stage's state with certainty; instead, the current decision only determines the probability distribution of the state at the next stage.
4. *Given the current state, the optimal decision for each of the remaining stages must not depend on previously reached states or previously chosen decisions.* This idea is known as the **principle of optimality.** In the context of Network Problem, the principle of optimality reduces to the following: Suppose the shortest path (call it *R*) from city 1 to city 10 is known to pass through city *i*. Then the portion of *R* that goes from city *i* to city 10 must be shortest path from city *i* to city 10. If this were not the case, then we could create a path from city 1 to city 10 that was shorter than *R* by appending a shortest path from city *i* to city 10 to the portion of *R* leading from city 1 to city *i*. This would create a path

from city 1 to city 10 that is shorter than R, thereby contradicting the fact that R is the shortest path from city 1 to city 10. For example, if the shortest path city 1 to city 10 must include a shortest path from city 2 than the shortest path from city 1 to city 10 must include a shortest path from city 2 to city 10 (2-5-8-10). This follows because any path from city 1 to city 10 that passes through city 2 and does not contain a shortest path from city 2 to city 10 will have a length of c12 + [something bigger than $f_2(2)$]. Of course, such a path cannot be shortest path from city 1 to city 10.

5. *If the states for the problem have been classified into one of T stages, there must be a recursion that relates the cost or reward earned during stages t, t+1, …, T to the cost or reward earned from stages t+1, t+2,…, T.* In essence, the recursion formalizes the working backward procedure. In Network Problem, our recursion could have been written as

$$f_t(i) = \min_j \{c_{ij} + f_{t+1}(j)\}$$

where j must be a stage t+1 city and $f_5(10)=0$.

6.4 An Inventory Problem

In this section, we illustrate how DP can be used to solve an inventory problem with the following characteristics:

1. Time is broken up into periods, the present period being period 1, the next period 2, and the final period T. At the beginning of period 1, the demand during each period is known.
2. At the beginning of each period, the firm must determine how many units should be produced. Production capacity during each period is limited.
3. Each period's demand must be met on time from inventory or current production. During any period in which production takes place, a fixed cost of production as well as a variable per-unit cost is incurred.
4. The firm has limited storage capacity. This is reflected by a limit on end-of-period inventory. A per-unit holding cost is incurred on each period's ending inventory.
5. The firm's goal is to minimize the total cost of meeting on time the demands for periods 1, 2, …, T.

In this model, the firm's inventory position is reviewed at the end of each period (say, at the end of each month), and then the production decision is made. Such a model is called a **periodic review model**. This model is in contrast to the continuous review models in which the firm knows its inventory position at all times and may place an order or begin production at any time.

If we exclude the setup cost for producing any units, the inventory problem just described is similar to the Sailco inventory problem that is solved by linear programming in Section 3.10. Here, we illustrate how DP can be used to determine a production schedule that minimizes the total cost incurred in an inventory problem that meets the preceding description.

Example 6.4. Inventory *(Winston 18.3, p.970)*

A company knows that the demand for its product during each of the next four months will be as follows: month 1, 1 unit; month 2, 3 units; month 3, 2 units; month 4, 4 units. At the beginning of each month, the company must determine how many units should be produced during current month. During a month in which any units are produced, a setup cost of $3 is incurred. In addition, there is a variable cost of $1 for every unit produced. At the end of each month, a holding cost of 50¢ per unit on hand is incurred. Capacity limitations allow a maximum of 5 units to be produced during each month. The size of the company's warehouse restricts the ending inventory for each month to 4 units at most. The company wants to determine a production schedule that will meet all demands on time and will minimize the sum of production and holding costs during the four months. Assume that 0 units are on hand at the beginning of the first month.

Answer

We can ensure that all demands are met on time be restricting each month's inventory to be nonnegative. To use DP to solve this problem, we need to identify the appropriate state, stage, and decision. The stage should be defined so that when one stage remains, the problem will be trivial to solve. If we are at the beginning of month 4, then the firm would meet demand at minimum cost by simply producing just enough units to ensure that (month 4 production) + (month 3 ending inventory) = (month 4 demand). Thus, when one month remains, the firm's problem is easy to solve. Hence, we let time represent the stage. In most DP problems, the stage has something to do with time.

At each stage (or month), the company must decide how many units to produce. To make this decision, the company need only know the inventory level at the beginning of the current month (or at the end of the previous month). Therefore, we let the state at any stage be the beginning inventory level.

Before writing a recursive relation that can be used to "build up" the optimal production schedule, we must first define $f_t(i)$ to be the minimum cost of meeting demands for months t, t+1, ..., 4 if i units are on hand at the beginning of month t. We define $c(x)$ to be the cost of producing x units during a period. Then $c(0)=0$, and for $x>0$, $c(x)=3+x$. Because of the limited storage capacity and the fact that all demand must be met on time, the possible states during

each period are 0, 1, 2, 3 and 4. Thus, we begin by determining $f_4(0)$, $f_4(1)$, $f_4(2)$, $f_4(3)$, and $f_4(4)$. Then we use this information to determine $f_3(0)$, $f_3(1)$, $f_3(2)$, $f_3(3)$ and $f_3(4)$. Then we determine $f_2(0)$, $f_2(1)$, $f_2(2)$, $f_2(3)$ and $f_2(4)$. Finally, we determine $f_1(0)$. Then we determine an optimal production level for each month. We define

$x_t(i)$ to be a production level during month t that minimizes the total cost during months t, t+1, ..., 4 if i units are on hand at the beginning of month t. We now begin to work backward.

MONTH 4 COMPUTATIONS

During month 4, the firm will produce just enough units to ensure that the month 4 demand of 4 units is met. This yields

$f_4(0)$=	cost of producing	4 – 0	units =	c(4)=	3 + 4=	$7	and	$x_4(0)$=	4 – 0	= 4
$f_4(1)$=	cost of producing	4 – 1	units =	c(3)=	3 + 3=	$6	and	$x_4(1)$=	4 – 1	= 3
$f_4(2)$=	cost of producing	4 – 2	units =	c(2)=	3 + 2=	$5	and	$x_4(2)$=	4 – 2	= 2
$f_4(3)$=	cost of producing	4 – 3	units =	c(1)=	3 + 1=	$4	and	$x_4(3)$=	4 – 3	= 1
$f_4(4)$=	cost of producing	4 – 4	units =	c(0)= $0			and	$x_4(4)$=	4 – 4	= 0

MONTH 3 COMPUTATIONS

How can we now determine $f_3(i)$ for i = 0, 1, 2, 3, 4? The cost $f_3(i)$ is the minimum cost incurred during months 3 and 4 if the inventory at the beginning of month 3 is i. For each possible production level x during month 3, total cost during months 3 and 4 is

$$(\tfrac{1}{2})(i + x - 2) + c(x) + f_4(i + x - 2) \qquad (1)$$

This follows because if x units are produced during month 3, the ending inventory for month 3 will be $i + x - 2$. Then the month 3 holding cost will be $(\tfrac{1}{2})(i + x - 2)$, and the month 3 production cost will be $c(x)$. Then we enter month 4 with $(i + x - 2)$ units on hand. Since we proceed optimally from this point onward (remember the principle of optimality), the cost for month 4 will be $f_4(i + x - 2)$. We want to choose the month 3 production level to minimize (1), so we write

$$f_3(i) = \min_x \left\{ \left(\frac{1}{2}\right)(i + x - 2) + c(x) + f_4(i + x - 2) \right\} \qquad (2)$$

In (2), x must be a member of {0, 1, 2, 3, 4, 5}, and x must satisfy $4 \geq i + x - 2 \geq 0$. This reflects the fact that the current month's demand must be met ($i + x - 2 \geq 0$), and ending inventory cannot exceed the capacity of 4 ($i + x - 2 \leq 4$). Recall that $x_3(i)$ is any value of x attaining $f_3(i)$. The computations for $f_3(0)$, $f_3(1)$, $f_3(2)$, $f_3(3)$, and $f_3(4)$:

i	x	(½) (i+x – 2) + $c(x)$	$f_4(i + x - 2)$	Total cost Months 3, 4	$f_3(i)$ $x_3(i)$
0	2	0 + 5 = 5	7	5 + 7 =12*	
0	3	0.5+6 = 6.5	6	6.5+6 = 12.5	$f_3(0)$=12
0	4	$1+7=8$	5	$8+5=13$	$x_3(0)$=2
0	5	1.5 + 8 = 9.5	4	9.5 +4 =13.5	
1	1	$0+4=4$	7	$4+7=11$	
1	2	0.5 + 5 = 5.5	6	5.5 + 6 = 11.5	$f_3(1)$=10
1	3	$1+6=7$	5	$7+5=12$	$x_3(1)$=5
1	4	1.5 + 7 = 8.5	4	8.5 + 4 = 12.5	
1	5	$2+8=10$	0	$10+0=10$ *	
2	0	$0+0=0$	7	$0+7=7$ *	
2	1	0.5+4 =4.5	6	4.5 + 6 =10.5	$f_3(2)$=7
2	2	$1+5=6$	5	$6+5=11$	$x_3(2)$=0
2	3	1.5 + 6 = 7.5	4	7.5 + 4 =11.5	
2	4	$2+7=9$	0	$9+0=9$	
3	0	0.5 + 0 = 0.5	6	0.5 + 6 = 6.5*	
3	1	$1+4=5$	5	$5+5=10$	$f_3(3)$=6.5
3	2	1.5 + 5 = 6.5	4	6.5 + 4 = 10.5	$x_3(3)$=0
3	3	$2+6=8$	0	$8+0=8$	
4	0	$1+0=1$	5	$1+5=6$ *	
4	1	1.5+4=5.5	4	5.5+4 =9.5	$f_3(4)$=6
4	2	$2+5=7$	0	$7+0=7$	$x_3(4)$=0

MONTH 2 COMPUTATIONS

We can now determine $f_2(i)$, the minimum cost incurred during months 2, 3, and 4 given that at the beginning of month 2, the on-hand inventory is i units. Suppose that month 2 production = x. Because month 2 demand is 3 units, a holding cost of (½) ($i + x - 3$) is incurred at the end of month 2. Thus, the total cost incurred during month 2 is (½) (i+x – 3) + $c(x)$. During months 3 and 4, we follow an optimal policy. Since month 3 begins with an inventory of ($i + x - 3$), the cost incurred during months 3 and 4 is $f_3(i + x - 3)$. In analogy to (2), we now write

$$f_2(i) = \min_x \left\{ \left(\frac{1}{2}\right)(i + x - 3) + c(x) + f_3(i + x - 3) \right\} \qquad (3)$$

where x must be a member of {0, 1, 2, 3, 4, 5} and x must satisfy $0 \le (i+x-3) \le 4$. The computations for $f_2(0)$, $f_2(1)$, $f_2(2)$, $f_2(3)$, and $f_2(4)$ are given in the following Table:

i	x	$(\frac{1}{2})(i+x-3)+c(x)$	$f_3(i+x-3)$	Total cost Months 2-4	$f_2(i)$ $x_2(i)$
0	3	$0+6=6$	12	$6+12=18$	$f_2(0)=16$
0	4	0.5+7=7.5	10	7.5 + 10=17.5	$x_2(0)=5$
0	5	$1+8=9$	7	$9+7=16$ *	
1	2	$0+5=5$	12	$5+12=17$	
1	3	0.5+6=6.5	10	6.5+10=16.5	$f_2(1)=15$
1	4	$1+7=8$	7	$8+7=15$ *	$x_2(1)=4$
1	5	1.5+8=9.5	6.5	9.5+6.5=16	
2	1	$0+4=4$	12	$4+12=16$	
2	2	0.5+5=5.5	10	5.5+10=15.5	$f_2(2)=14$
2	3	$1+6=7$	7	$7+7=14$ *	$x_2(2)=3$
2	4	1.5+7=8.5	6.5	8.5+6.5=15	
2	5	$2+8=10$	6	$10+6=16$	
3	0	$0+0=0$	12	$0+12=12$ *	
3	1	0.5+4=4.5	10	4.5+10=14.5	$f_2(3)=12$
3	2	$1+5=6$	7	$6+7=13$	$x_3(3)=0$
3	3	1.5+6=7.5	6.5	7.5+6.5=14	
3	4	$2+7=9$	6	$9+6=15$	
4	0	0.5+0=0.5	10	0.5+10=10.5*	
4	1	$1+5=6$	7	$5+7=12$	$f_2(4)=10.5$
4	2	1.5+5=6.5	6.5	6.5+6.5=13	$x_2(4)=0$
4	3	$2+6=8$	6	$8+6=14$	

MONTH 1 COMPUTATIONS

$f_1(i)$'s can be determined via the following recursive relation:

$$f_1(i)=\min_x\left\{\left(\frac{1}{2}\right)(i+x-1)+c(x)+f_2(i+x-1)\right\} \qquad (4)$$

where x must be a member of {0, 1, 2, 3, 4, 5} and x must satisfy $0 \le (i+x-1) \le 4$. Since the inventory at the beginning of month 1 is 0 units, we actually need only determine $f_1(0)$ and $x_1(0)$.

i	x	(½) (i+x – 1) + c(x)	$f_2(i + x - 1)$	Total cost	$f_1(i)$ $x_1(i)$
0	1	$0+4=4$	16	$4+16=20$ *	
0	2	0.5+5=5.5	15	5.5+15=20.5	$f_1(0)=20$ $x_1(0)=1$
0	3	$1+6=7$	14	$7+14=21$	
0	4	1.5+7=8.5	12	8.5+12=20.5	
0	5	$2+8=10$	10.5	10+10.5=20.5	

Optimal Solution and Report

We can now determine a production schedule that minimizes the total cost of meeting the demand for all four months on time. Since our initial inventory is 0 units, the minimum cost for the four months will be $f_1(0)$=\$20. To attain $f_1(0)$, we must produce $x_1(0)$=1 unit during month 1. Then the inventory at the beginning of month 2 will be 0 + 1 – 1 = 0. Thus, in month 2, we should produce $x_2(0)$=5 units. Then at the beginning of month 3, our beginning inventory will be 0 + 5 – 3 = 2. Hence, during month 3, we need to produce $x_3(2)$=0 units. Then month 4 will begin with 2 – 2 + 0 = 0 units on hand. Thus, $x_4(0)$=4 units should be produced during month 4.

In summary, the optimal production schedule incurs a total cost of \$20 and produces 1 unit during month 1, 5 units during month 2, 0 units during month 3, and 4 units during month 4.

6.5 Resource Allocation Problems

Resource allocation problems, in which limited resources must be allocated among several activities, are often solved by DP. Recall that we have solved such problems by linear programming (for instance, the Giapetto problem). To use linear programming to do resource allocation, three assumptions must be made:

1. The amount of a resource assigned to an activity may be any nonnegative number.
2. The benefit obtained from each activity is proportional to the amount of resource assigned to this activity.
3. The benefit obtained from more than one activity is the sum of the benefits from the individual activities.

Even if assumptions 1 and 2 do not hold, DP can be used to solve resource allocation problems efficiently when assumption 3 is valid and when the amount of the resource allocated to each activity is a member of a finite set.

Example 6.5. Resource Allocation *(Winston 18.4, p.975)*

Finco has \$6000 to invest, and three investments are available. If d_j dollars (in thousands) are invested in investment *j,* then a net present value (in thousands) of $r_j(d_j)$ is obtained, where the $r_j(d_j)$'s are as follows:

$$r_1(d_1) = 7d_1 + 2 \qquad (d_1 > 0)$$
$$r_2(d_2) = 3d_2 + 7 \qquad (d_2 > 0)$$
$$r_3(d_3) = 4d_3 + 5 \qquad (d_3 > 0)$$
$$r_1(0) = r_2(0) = r_3(0) = 0$$

The amount placed in each investment must be an exact multiple of \$1000. To maximize the net present value obtained from these investments, how should Finco allocate the \$6000?

Answer

The return on each investment is not proportional to the amount invested in it [for example, $16=r_1(2)\neq 2r_1(1)=18$]. Thus, linear programming cannot be used to find an optimal solution to this problem.

Mathematically, Finco's problem may be expressed as

max $\{r_1(d_1) + r_2(d_2) + r_3(d_3)\}$

s.t. $d_1 + d_2 + d_3 = 6$

$d_j > 0$ and integer (j=1,2,3)

Of course, if $r_j(d_j)$'s were linear, then we would have a knapsack problem.

To formulate Finco's problem as a DP problem, we begin by identifying the stage. As in the inventory and shortest-route examples, the stage should be chosen so that when one stage remains the problem is easy to solve. Then, given that the problem has been solved for the case where one stage remains, it should be easy to solve the problem where two stages remain, and so forth. Clearly, it would be easy to solve when only one investment was available, so we define stage t to represent a case where funds must be allocated to investments t, t+1, ..., 3.

For a given stage, what must we know to determine the optimal investment amount? Simply how much money is available for investments t, t+1, ..., 3. Thus, we define the state at any stage to be the amount of Money (in thousands) available for investments t, t+1, ..., 3. We can never have more than \$6000 available, so the possible states at any stage are 0, 1, 2, 3, 4, 5, and 6. We define $f_t(d_t)$ to be the maximum net present value (NPV) that can be obtained by investing d_t thousand dollars in investments t, t+1,..., 3. Also define $x_t(d_t)$ to be the amount that should be invested in investment t to attain $f_t(d_t)$. We start to work backward by computing $f_3(0)$, $f_3(1)$, ..., $f_3(6)$ and then determine $f_2(0)$, $f_2(1)$,...,$f_2(6)$. Since \$6000 is available for investment in investments 1, 2, and 3, we terminate our computations by computing $f_1(6)$. Then we retrace our steps and determine the amount that should be allocated to each

investment (just as we retraced our steps to determine the optimal production level for each month in inventory example).

STAGE 3 COMPUTATIONS

We first determine $f_3(0)$, $f_3(1)$,…, $f_3(6)$. We see that $f_3(d_3)$ is attained by investing all available money (d_3) in investment 3. Thus,

$f_3(0) = 0$ $\quad x_3(0) = 0$

$f_3(1) = 9$ $\quad x_3(1) = 1$

$f_3(2) = 13$ $\quad x_3(2) = 2$

$f_3(3) = 17$ $\quad x_3(3) = 3$

$f_3(4) = 21$ $\quad x_3(4) = 4$

$f_3(5) = 25$ $\quad x_3(5) = 5$

$f_3(6) = 29$ $\quad x_3(6) = 6$

STAGE 2 COMPUTATIONS

To determine $f_2(0)$, $f_2(1)$,...,$f_2(6)$, we look at the possible amounts that can be placed in investment 2. To find $f_2(d_2)$, let x_2 be the amount invested in investment 2. Then an NPV of $r_2(x_2)$ will be obtained from investment 2, and an NPV of $f_3(d_2–x_2)$ will be obtained from investment 3 (remember the principle of optimality). Since x_2 should be chosen to maximize the net present value earned from investments 2 and 3, we write

$$f_2(d_2) = \max_{x_2}\{r_2(x_2) + f_3(d_2 - x_2)\} \qquad (5)$$

where x_2 must be a member of $\{0,1,...,d_2\}$. The computations for $f_2(0)$, $f_2(1)$, ..., $f_2(6)$ and $x_2(0)$, $x_2(1)$, ..., $x_2(6)$ are given in the following Table:

d_2	x_2	$r_2(x_2)$	$f_3(d_2–x_2)$	NPV from investments 2, 3	$f_2(d_2)$ $x_2(d_2)$
0	0	0	0	0*	$f_2(0)=0$ $x_2(0)=0$
1	0	0	9	9	$f_2(1)=10$
1	1	10	0	10*	$x_2(1)=1$
2	0	0	13	13	
2	1	10	9	19*	$f_2(2)=19$ $x_2(2)=1$
2	2	13	0	13	
3	0	0	17	17	
3	1	10	13	23*	$f_2(3)=23$
3	2	13	9	22	$x_2(3)=1$
3	3	16	0	16	
4	0	0	21	21	$f_2(4)=27$

4	1	10	17	27*	$x_2(4)=1$
4	2	13	13	26	
4	3	16	9	25	
4	4	19	0	19	
5	0	0	25	25	
5	1	10	21	31*	
5	2	13	17	30	$f_2(5)=31$
5	3	16	13	29	$x_2(5)=1$
5	4	19	9	28	
5	5	22	0	22	
6	0	0	29	29	
6	1	10	25	35*	
6	2	13	21	34	$f_2(6)=35$
6	3	16	17	33	$x_2(6)=1$
6	4	19	13	32	
6	5	22	9	31	
6	6	25	0	25	

STAGE 1 COMPUTATIONS

Following (5), we write

$$f_1(6) = \max_{x_1}\{r_1(x_1) + f_2(6 - x_1)\}$$

where x_1 must be a member of {0, 1, 2, 3, 4, 5, 6}. The computations for $f_1(6)$ are given in the following Table:

d_1	x_1	$r_1(x_1)$	$f_2(6–x_1)$	NPV from investments 1-3	$f_1(6)$ $x_1(6)$
6	0	0	35	35	
6	1	9	31	40	
6	2	16	27	43	
6	3	23	23	46	$f_1(6)=49$
6	4	30	19	49*	$x_1(6)=4$
6	5	37	10	47	
6	6	44	0	44	

Optimal Solution and Report

Since $x_1(6)=4$, Finco invests \$4000 in investment 1. This leaves \$6000 – \$4000 = \$2000 for investments 2 and 3. Hence, Finco should invest $x_2(2)$=\$1000 in investment 2. Then \$1000 is left for investment 3, so Finco chooses to invest $x_3(1)$=\$1000 in investment 3.

Finco can attain a maximum net present value of $f_1(6)$=\$49000 by investing \$4000 in investment 1, \$1000 in investment 2, and \$1000 in investment 3.

6.5.1 Generalized Resource Allocation Problem

We now consider a generalized version of resource allocation example. Suppose we have w units of a resource available and T activities to which the resource can be allocated. If activity t is implemented at a level x_t (we assume x_t must be a nonnegative integer), then $g_t(x_t)$ units of the resource are used by activity t, and a benefit $r_t(x_t)$ is obtained. The problem of determining the allocation of resources that maximizes total benefit subject to the limited resource availability may be written as

$$\max \sum_{t=1}^{t=T} r_t(x_t)$$
$$\text{s.t.} \sum_{t=1}^{t=T} g_t(x_t) \le w \qquad (6)$$

where x_t must be a member of {0, 1, 2,...}.

Some possible interpretations of $r_t(x_t)$, $g_t(x_t)$, and w are given in the Table given below:

$r_t(x_t)$	$g_t(x_t)$	w
Benefit from placing x_t type t items in a knapsack	Weight of x_t type t items	Maximum weight that knapsack can hold
Grade obtained in course t if we study course t for x_t hours per week	Number of hours per week x_t spent studying course t	Total number of study hours available each week
Sales of a product in region t if x_t sales reps are assigned to region t	Cost of assigning x_t sales reps to region t	Total sales force budget
Number of fire alarms per week responded to within one minute if precinct t is assigned x_t engines	Cost per week of maintaining x_t fire engines in precinct t	Total weekly budget for maintaining fire engines

To solve the given LP by DP, define $f_t(d)$ to be the maximum benefit that can be obtained from activities t, t+1, ..., T if d units of the resource may be allocated to activities t, t+1, ..., T. We may generalize the recursions of resource allocation example to this situation by writing

$$f_{T+1}(d)=0 \quad \text{for all } d$$
$$f_t(d)=\max_{x_t} \{r_t(x_t) + f_{t+1}(d - g_t(x_t)) \qquad (7)$$

where x_t must be a nonnegative integer satisfying $g_t(x_t) \le d$. Let $x_t(d)$ be any value of x_t that attains $f_t(d)$. To use the formula to determine an optimal allocation of resources to activities 1,

2, ..., T, we begin by determining all $f_T(\cdot)$ and $x_T(\cdot)$. Then we use (7) to determine all $f_{T-1}(\cdot)$ and $x_{T-1}(\cdot)$, continuing to work backward in this fashion until all $f_2(\cdot)$ and $x_2(\cdot)$ have been determined. To wind things up, we now calculate $f_1(w)$ and $x_1(w)$. Then we implement activity 1 at a level $x_1(w)$. At this point, we have $w-g_1(x_1(w))$ units of the resource available for activities 2, 3, ..., T. Then activity 2 should be implemented at a level of $x_2[w-g_1(x_1(w))]$. We continue in this fashion until we have determined the level at which all activities should be implemented.

6.5.2 Solution of Knapsack Problems by DP

The Knapsack problem can be solved by dynamic programming.

Example 6.6. Knapsack *(Winston 18.4, p.979)*

Suppose a 10-lb knapsack is to be filled with items listed in the following Table. To maximize total benefit, how should the knapsack be filled?

Item	Weight	Benefit
1	4	11
2	3	7
3	5	12

Answer

We have $r_1(x_1) = 11x_1$, $r_2(x_2) = 7x_2$, $r_3(x_3) = 12x_3$, $g_1(x_1) = 4x_1$, $g_2(x_2) = 3x_2$, and $g_3(x_3) = 5x_3$. Define $f_t(d)$ to be the maximum benefit that can be earned from a d-pound knapsack that is filled with items of Type t, t+1, ..., 3.

STAGE 3 COMPUTATIONS

Now (7) yields

$$f_3(d) = \max_{x_3}\{12x_3\}$$

where $5x_3 \le d$ and x_3 is a nonnegative integer. These yields

$f_3(10) = 24$

$f_3(5) = f_3(6) = f_3(7) = f_3(8) = f_3(9) = 12$

$f_3(0) = f_3(1) = f_3(2) = f_3(3) = f_3(4) = 0$

$x_3(10) = 2$

$x_3(9) = x_3(8) = x_3(7) = x_3(6) = x_3(5) = 1$

$x_3(0) = x_3(1) = x_3(2) = x_3(3) = x_3(4) = 0$

STAGE 2 COMPUTATIONS

Now (7) yields

$$f_2(d) = \max_{x_2}\{7x_2 + f_3(d-3x_2)\}$$

where x_2 must be a nonnegative integer satisfying $3x_2 \le d$. We now obtain

$$f_2(10)=\max\begin{cases}7(0)+f_3(10)=24\ * & x_2=0\\ 7(1)+f_3(7)=19 & x_2=1\\ 7(2)+f_3(4)=14 & x_2=2\\ 7(3)+f_3(1)=21 & x_2=3\end{cases}$$

Thus, $f_2(10)=24$ and $x_2(10)=0$

$$f_2(9)=\max\begin{cases}7(0)+f_3(9)=12 & x_2=0\\ 7(1)+f_3(8)=19 & x_2=1\\ 7(2)+f_3(3)=14 & x_2=2\\ 7(3)+f_3(0)=21\ * & x_2=3\end{cases}$$

Thus, $f_2(9)=21$ and $x_2(9)=3$

$$f_2(8)=\max\begin{cases}7(0)+f_3(8)=12 & x_2=0\\ 7(1)+f_3(5)=19\ * & x_2=1\\ 7(2)+f_3(2)=14 & x_2=2\end{cases}$$

Thus, $f_2(8)=19$ and $x_2(8)=1$

$$f_2(7)=\max\begin{cases}7(0)+f_3(7)=12 & x_2=0\\ 7(1)+f_3(4)=7 & x_2=1\\ 7(2)+f_3(1)=14\ * & x_2=2\end{cases}$$

Thus, $f_2(7)=14$ and $x_2(7)=2$

$$f_2(6)=\max\begin{cases}7(0)+f_3(6)=12 & x_2=0\\ 7(1)+f_3(3)=7 & x_2=1\\ 7(2)+f_3(0)=14\ * & x_2=2\end{cases}$$

Thus, $f_2(6)=14$ and $x_2(6)=2$

$$f_2(5)=\max\begin{cases}7(0)+f_3(5)=12\ * & x_2=0\\ 7(1)+f_3(2)=7 & x_2=1\end{cases}$$

Thus, $f_2(5)=12$ and $x_2(5)=0$

$$f_2(4)=\max\begin{cases}7(0)+f_3(4)=0 & x_2=0\\ 7(1)+f_3(1)=7\ * & x_2=1\end{cases}$$

Thus, $f_2(4)=7$ and $x_2(4)=1$

$$f_2(3)=\max\begin{cases}7(0)+f_3(3)=0 & x_2=0\\ 7(1)+f_3(0)=7\ * & x_2=1\end{cases}$$

Thus, $f_2(3)=7$ and $x_2(3)=1$

$$f_2(2)=\max\begin{cases}7(0)+f_3(2)=0 & x_2=0\end{cases}$$

Thus, $f_2(2)=0$ and $x_2(2)=0$

$$f_2(1)=\max\begin{cases}7(0)+f_3(1)=0 & x_2=0\end{cases}$$

Thus, $f_2(1)=0$ and $x_2(1)=0$

$$f_2(0)=7(0)+f_3(0)=0 \qquad x_2=0$$

Thus, $f_2(0)=0$ and $x_2(0)=0$

STAGE 1 COMPUTATIONS

Finally, we determine $f_1(10)$ from

$$f_1(10)=\max\begin{cases}11(0)+f_2(10)=24 & x_1=0\\ 11(1)+f_2(6)=25^* & x_1=1\\ 11(2)+f_2(2)=22 & x_1=2\end{cases}$$

Optimal Solution and Report

We have $f_1(10) = 25$ and $x_1(10) = 1$. Hence, we should include one Type 1 item in the knapsack. Then we have 10 – 4 = 6 lb left for Type 2 and Type 3 items, so we should include $x_2(6) = 2$ Type 2 items. Finally we have 6 – 2(3) = 0 lb left for Type 3 items, and we include $x_3(0)=0$ Type 3 items. In summary, the maximum benefit that can be gained from a 10-lb knapsack is $f_3(10)=25$.

To obtain a benefit of 25, one Type 1 and two Type 2 items should be included.

Alternative formulation to Knapsack Problem

If we define $g(w)$ as maximum benefit from w weight knapsack, the following recursion can be used to solve the knapsack problem:

$$g(w) = \max_j\{b_j + g(w - w_j)\} \qquad (8)$$

where j: items, b_j: benefit of item j, w_j: weight of item j .

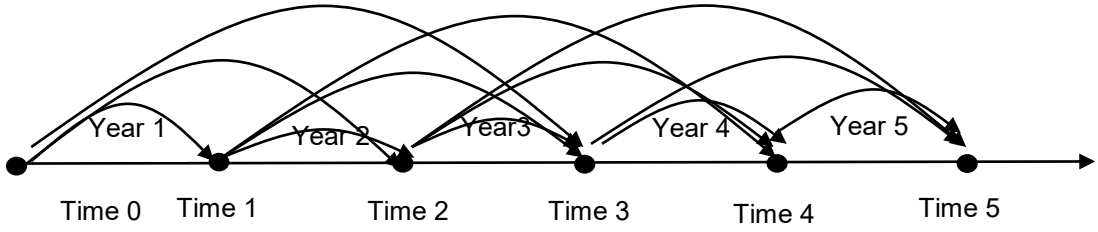
6.6 Equipment Replacement Problems

Many companies and customers face the problem of determining how long a machine should be utilized before it should be traded in for a new one. Problems of this type are called equipment-replacement problems and can often be solved by DP.

Example 6.7. Equipment Replacement *(Winston 18.5, p.985)*

An auto repair always needs to have an engine analyzer available. A new engine analyzer cost \$1,000. The cost m_i of maintaining an engine analyzer during its ith year of operation is

as follows: m_1 = \$60, m_2 = \$80, m_3 = \$120. An analyzer may be kept for 1, 2, or 3 years; after i years of use (i=1, 2, 3), it may be traded in for a new one. If an i-year-old engine analyzer is traded in' a salvage value s_i is obtained, where s_1=\$800, s_2=\$600, s_3=\$500. Given that a new machine must be purchased now (time 0; see the following figure), the shop wants to determine a replacement and trade-in policy that minimizes net costs =(maintenance costs) + (replacement costs) – (salvage value received) during the next 5 years.



Answer

We note that after a new machine is purchased, the firm must decide when the newly purchased machine should be traded in for a new one. With this in mind, we define $g(t)$ to be the minimum net cost incurred from time t until time 5 (including the purchase cost and salvage value for the newly purchased machine) given that a new machine has been purchased at time t. We also define c_{tx} to be the net cost (including purchase cost and salvage value) of purchasing a machine at time t and operating it until time x. Then the appropriate recursion is

$$g(t) = \min_x \{c_{tx} + g(x)\} \qquad (t = 0,\ 1,\ 2,\ 3,\ 4) \qquad (9)$$

where x must satisfy the inequalities $t+1 \le x \le t+3$ and $x \le 5$. Because the problem is over at time 5, no cost incurred from time 5 onward, so we may write $g(5)=0$.

To justify (9), note that after a new machine is purchased at time t, we must decide when to replace the machine. Let x be the time at which the replacement must be after time t but within 3 years of time t. This explains the restriction that $t+1 \le x \le t+3$. Since the problem ends at time 5, we must also have $x \le 5$. If we choose to replace the machine at time x, then what will be the cost from time t to time 5? Simply the sum of the cost incurred from the purchase of the machine to the sale of the machine at time x (which is by definition c_{tx}) and the total cost incurred from time x to time 5 (given that a new machine has just been purchased at time x). By the principle of optimality, the latter cost is, of course, $g(x)$. Hence, if we keep the machine that was purchased at time t until time x, then from time t to time 5, we incur a cost of $c_{tx}+ g(x)$. Thus, x should be chosen to minimize this sum, and this is exactly what (9) does. We have assumed that maintenance costs, salvage value, and purchase price remain unchanged over time, so each c_{tx} will depend only on how long the machine is kept; that is, each c_{tx} depends only on x-t. More specifically,

c_{tx} =\$1000 + m_1 + ... + m_{x-t} – s_{x-t}

This yields

$$c_{01}=c_{12}=c_{23}=c_{34}=c_{45}=1000+60-800=\$260$$
$$c_{02}=c_{13}=c_{24}=c_{35}=1000+60+80-600=\$540$$
$$c_{03}=c_{14}=c_{25}=1000+60+80+120-500=\$760$$

We begin computing $g(4)$ and work backward until we have computed $g(0)$. Then we use our knowledge of the values of x attaining $g(0)$, $g(1)$, $g(2)$, $g(3)$ and $g(4)$ to determine the optimal replacement strategy. The calculations follow.

At time 4, there is only one sensible decision (keep the machine until time 5 and sell it for its salvage value), so we find

$$g(4)=c_{45}+g(5)=260+0=\$260\ *$$

Thus, if a new machine is purchased at time 4, it should be traded in at time 5.

If a new machine is purchased at time 3, we keep it until time 4 or time 5. Hence,

$$g(3)=\min\begin{cases}c_{34}+g(4)=260+260=\$520\ *\\ c_{35}+g(5)=540+0=\$540\end{cases}$$

Thus, if a new machine is purchased at time 3, we should trade it in at time 4.

If a new machine is purchased at time 2, we trade it in at time 3, 4, or 5. This yields

$$g(2)=\min\begin{cases}c_{23}+g(3)=260+520=\$780\\ c_{24}+g(4)=540+260=\$800\\ c_{25}+g(5)=\$760\ *\end{cases}$$

Thus, if we purchase a new machine at time 2, we should keep it until time 5 and then trade it in.

If a new machine is purchased at time 1, we trade it in at time 2, time 3, or time 4. Then

$$g(1)=\min\begin{cases}c_{12}+g(2)=260+760=1020\ *\\ c_{13}+g(3)=540+520=1060\\ c_{14}+g(4)=760+260=1020\ *\end{cases}$$

Thus, if we purchase a new machine at time 1, it should be traded in at time 2 or 4.

The new machine that was purchased at time 0 may be traded in at time 1, time 3, or time 4. Thus,

$$g(0)=\min\begin{cases}c_{01}+g(1)=260+1020=1280\ *\\ c_{02}+g(2)=540+760=1300\\ c_{03}+g(3)=760+520=1280\ *\end{cases}$$

Optimal Solution and Report

Thus, the new machine purchased at time 0 should be replaced at time 1 or time 3. Let's arbitrarily choose to replace the time 0 machine at time 1. Then the new time 1 machine may be traded in at time 2 or time 4. Again, we make an arbitrary choice and replace the time 1 machine at time 2. Then the time 2 machine should be kept until time 5, when it is sold for salvage value. With this replacement policy, we will incur a net cost of $g(0)$=\$1,280.

The reader should verify that the following replacement policies are also optimal: trading in at times 1, 4, and 5 as well as trading in at times 3, 4, and 5.

Review

We have assumed that all costs remain stationary over time. This assumption was made solely to simplify the computation of the c_{tx}'s. If we had relaxed the assumption of stationary costs, then the only complication would have been that the c_{tx}'s would have been messier to compute. We also note that if a short planning horizon is used, the optimal replacement policy may be extremely sensitive to the length of the planning horizon. Thus, more meaningful results can be obtained by using a longer planning horizon.

6.7 Formulating DP Recursions

In many DP problems (such as the inventory and shortest path examples), a given stage simply consists of all the possible states that the system can occupy at that stage. If this is the case, then the DP recursion (for a min problem) can often be written in the following form:

$$f_t(i)=\min\{(\text{cost during stage } t) + f_{t+1}(\text{new state at stage } t+1)\} \qquad (10)$$

where the minimum in (10) is over all decisions that are allowable, or feasible, when the state at stage t is i. In (10), $f_t(i)$ is the minimum cost incurred from stage t to the end of the problem (say, the problem ends at stage T), given that at stage t the state is i.

Equation (10) reflects the fact that the minimum cost incurred from stage t to the end of the problem must be attained by choosing at stage t an allowable decision that minimizes the sum of the costs incurred during the current stage (stage t) plus the minimum cost that can be incurred from stage $t+1$ to the end of the problem. Correct formulation of a recursion of the form (10) requires that we identify three important aspects of the problem:

1. *The set of decisions that is allowable, or feasible, for the given state and stage.* Often, the set of feasible decisions depends on both t and i. For instance, in the inventory example, let

   d_t = demand during month t

   i_t = inventory at beginning of month t

   In this case, the set of allowable month t decisions (let x_t represent an allowable production level) consists of the members of {0, 1, 2, 3, 4, 5} that satisfy $0 \le (i_t + x_t - d_t) \le$

4. Note how the set of allowable decisions at time t depends on the stage t and the state at time t, which is i_t.

2. *We must specify how the cost during the current time period (stage t) depends on the value of t, the current state, and the decision chosen at stage t.* For instance, in the inventory example, suppose a production level x_t is chosen during month t. Then the cost during month t is given by $c(x_t) + (½)(i_t + x_t - d_t)$.
3. *We must specify how the state at stage t + 1 depends on the value of t, the state at stage t, and the decision chosen at stage t.* Again referring to the inventory example, the month t + 1 state is $i_t + x_t - d_t$.

If you have properly identified the state, stage, and decision, then aspects 1 – 3 shouldn't be too hard to handle. A word of caution, however: Not all recursions are of the form of (10).

Example 6.8. A Fishery *(Winston 18.6, p.990)*

The owner of a lake must decide how many bass to catch and sell each year. If she sells x bass during year t, then a revenue of $r(x)$ is earned. The cost of catching x bass during a year is a function $c(x, b)$ of the number of bass caught during the year and of b, the number of bass in the lake at the beginning of the year. Of course, bass do reproduce. To model this, we assume that the number of bass in the lake at the beginning of a year is 20% more than the number of bass left in the lake at the end of the previous year. Develop a DP recursion that can be used to maximize the owner's net profits over a T-year horizon.

Answer

In problems where decisions must be made at several points in time, there is often a trade-off of current benefits against future benefits. For example, we could catch may bass early in the problem, but then the lake would be depleted in later years, and there would be very few bass to catch. On the other hand, if we catch very few bass now, we won't make much money early, but we can make a lot of money near the end of the horizon. In intertemporal optimization problems, DP is often used to analyze these complex trade-offs.

At the beginning of year T, the owner of the lake needs no worry about the effect that the capture of bass will have on the future population of the lake. (At time T, there is no future!) So at the beginning of year T, the problem is relatively easy to solve. For this reason, we let time be the stage. At each stage, the owner of the lake must decide how many bass to catch. We define x_t to be the number of bass caught during year t. To determine an optimal value of x_t, the owner of the lake need only know the number of bass (call it b_t) in the lake at the beginning of the year t. Therefore, the state at the beginning of year t is b_t.

We define $f_t(b_t)$ to be the maximum net profit that can be earned from bass caught during years t, t+1,...,T given that b_t bass are in the lake at the beginning of year t. We may dispose of aspects 1 – 3 of the recursion.

1. What are the allowable decisions? During any year, we can't catch more bass than there are in the lake. Thus, in each state and for all t, $0 \le x_t \le b_t$ must hold.
2. What is the net profit earned during year t? If x_t bass are caught during a year that begins with b_t bass in the lake, then the net profit is $r(x_t)$-$c(x_t, b_t)$.
3. What will be the state during year t+1? At the end of year t, there will be $b_t - x_t$ *bass* in the lake. By the beginning of year t + 1, these bass will have multiplied by 20%. This implies that at the beginning of year t+1, $1.2(b_t - x_t)$ bass will be in the lake. Thus, the year t+1 state will be $1.2(b_t - x_t)$.

We can now use (10) to develop the appropriate recursion. After year T, there are no future profits to consider, so

$$f_T(b_T) = \max_{x_T}\{r_T(x_T) - c(x_T, b_T)\}$$

where $0 \le x_T \le b_T$. Applying (10), we obtain

$$f_t(b_t) = \max\{r_t(x_t) - c(x_t, b_t) + f_{t+1}(1.2(b_t - x_t))\} \quad (11)$$

where $0 \le x_t \le b_t$.

6.7.1 Incorporating the Time Value of Money into DP Formulations

A weakness of the current formulation is that profits received during later years are weighted the same as profits received during earlier years. As mentioned in the discussion of discounting, later profits should be weighted less than the earlier profits. Suppose that for some β<1, $1 received in at the beginning of year t+1 is equivalent to β dollars received at the beginning of year t. We can incorporate this idea into the DP recursion by replacing (12) with

$$f_t(b_t) = \max\{r_t(x_t) - c(x_t, b_t) + \beta f_{t+1}(1.2(b_t - x_t))\} \quad (12)$$

where $0 \le x_t \le b_t$.

Example 6.9. Power Plant *(Winston 18.6, p.991)*

An electric power utility forecasts that r_t kilowatt-hours (kWh) of generating capacity will be needed during year t (the present year is year 1). Each year, the utility must decide by how much generating capacity should be expanded. It costs $c_t(x)$ dollars to increase generating capacity by x kWh during year t. Since it may be desirable to reduce capacity, x need not be nonnegative. During each year, 10% of the old generating capacity becomes obsolete and unusable (capacity does not become obsolete during its first year of operation). It costs the

utility $m_t(i)$ dollars to maintain i units of capacity during year t. At the beginning of year 1, 100,000 kWh of generating capacity are available. Formulate a DP recursion that will enable the utility to minimize the total cost of meeting power requirements for the next T years.

Answer

Again, we let the time be the stage. At the beginning of year t, the utility must determine the amount of capacity (call it x_t) to add during year t. To choose x_t properly, all the utility needs to know is the amount of available capacity at the beginning of year t (call it i_t). Hence, we define the state at the beginning of year t to be the current capacity level. We may now dispose of aspects 1-3 of the formulation.

1. What values of x_t are feasible? To meet year t's requirement of r_t, we must have $i_t+x_t \geq r_t$, or $x_t \geq r_t - i_t$. So the feasible x_t's are those values of x_t satisfying $x_t \geq r_t - i_t$.
2. What cost is incurred during year t? If x_t kWh are added during a year that begins with i_t kWh of available capacity, then during year t, a cost $c_t(x_t)+m_t(x_t+i_t)$ is incurred.
3. What will be the state at the beginning of year t+1? At the beginning of year t+1, the utility will have $0,9i_t+x_t$ kWh of old capacity plus the x_t kWh that have been added during year t. thus, the state at the beginning of year t+1 will be $0.9i_t+x_t$.

$f_t(i_t)$, is the minimum cost incurred by the utility during years t, t+1,...,T.

$$f_T(i_T) = \min_{x_T}\{c_T(x_T) + m_T(i_T + x_T)\} \qquad (13)$$

$x_T \geq r_T - i_T$

For $t<T$,

$$f_t(i_t) = \min_{x_t}\{c_t(x_t) + m_t(i_t + x_t) + f_{t+1}(0.9i_t + x_t)\} \qquad (14)$$

$x_t \geq r_t - i_t$

Example 6.10. Wheat Sale *(Winston 18.6, p.992)*

Farmer Jones now possesses $5000 in cash and 1000 bushels of wheat. During month t, the price of wheat is p_t. During each month, the must decide how many bushels of wheat to buy (or sell). There are three restrictions on each month's wheat transactions: (1) During any month the amount of money spent on wheat cannot exceed the cash on hand at the beginning of the month; (2) during any month, he cannot sell more wheat than he has at the beginning of the month; and (3) because of limited warehouse capacity, the ending inventory of wheat for each month cannot exceed 1000 bushels.

Show how DP can be used to maximize the amount of cash that farmer Jones has on hand at the end of six months.

Answer

Again, we let time be the stage. At the beginning of month t (the present is the beginning of month 1), farmer Jones must decide by how much to change the amount of wheat on hand. We define Δw_t to be the change in farmer Jones's wheat position during month t; $\Delta w_t \geq 0$ corresponds to a month t wheat purchase, and $\Delta w_t \leq 0$ corresponds to a month t sale of wheat. To determine an optimal value for Δw_t, we must know two things: the amount of wheat on hand at the beginning of month t (call it w_t) and the cash on hand at the beginning of month t, (call this c_t). We define $f_t(c_t, w_t)$ to be the maximum cash that farmer Jones can obtain at the end of month 6, given that farmer Jones has c_t dollars and w_t bushels of wheat at the beginning of month t. We now discuss aspects 1–3 of the formulation.

ASPECT1

$p_t(\Delta w_t) \leq c_t$ ensures that we won't run out of money at the end of month t.

$\Delta w_t \geq -w_t$ ensures that during month t, we will not sell more wheat than we had at the beginning of month t.

$w_t + \Delta w_t \leq 1000$ ensures that we will end month t with at most 1000 bushels of wheat.

Putting these three restrictions together, we see that

$$-w_t \leq \Delta w_t \leq \min\left\{\frac{c_t}{p_t}, 1000 - w_t\right\}$$

will ensure that restrictions 1-3 are satisfied during month t.

ASPECT 2

Since farmer Jones wants to maximize his cash on hand at the end of month 6, no benefit is earned during months 1 through 5. In effect, during months 1-5, we are doing bookkeeping to keep track of farmer Jones's position. Then, during month 6, we turn all of farmer Jones's assets into cash.

ASPECT 3

If the current state is (c_t, w_t) and farmer Jones changes his month t wheat position by an amount of Δw_t, what will be the new state at the beginning of month t+1? Cash on hand will increase by $-(\Delta w_t)p_t$, and farmer Jones's wheat position will increase by Δw_t. Hence, the month t+1 state will be

$$[c_t - (\Delta w_t)p_t, w_t + \Delta w_t].$$

At the end of month 6, farmer Jones should convert his month 6 wheat into cash by selling all of it. This means $\Delta w_6 = -w_t$.

$$f_6(c_6, w_6) = c_6 + w_6 p_6 \qquad (15)$$

for t<6,

$$f_t(c_t, w_t) = \max_{\Delta w_t}\{0 + f_{t+1}(c_t - (\Delta w_t)p_t, w_t + \Delta w_t)\} \qquad (16)$$

and $-w_t \le \Delta w_t \le \min\left\{\frac{c_t}{p_t}, 1000 - w_t\right\}$

Example 6.11. Refinery Capacity *(Winston 18.6, p.993)*

Sunco Oil needs to enough refinery capacity to refine 5000 barrels of oil per day and 10,000 barrels of gasoline per day. Sunco can build refinery capacity at four locations. The cost of building a refinery at site t that has the capacity to refine x barrels of oil per day and y barrels of gasoline per day is $c_t(x, y)$. Use DP to determine how much capacity should be located at each site.

Answer

If Sunco had only one possible refinery site, then the problem would be easy to solve. Sunco could solve a problem in which there were two possible refinery sites, and finally, a problem in which there were four refinery sites. For this reason, we let the stage represent the number of available oil sites. At any stage, Sunco must determine how much oil and gas capacity should be built at the available sites. We now define $f_t(o_t, g_t)$ to be the minimum cost of building o_t barrels per day of oil refinery capacity and g_t barrels per day of gasoline refinery capacity at sites t, t+1,...,4.

To determine $f_4(o_4, g_4)$, note that if only site 4 is available, Sunco must build a refinery at site 4 with o_4 barrels of oil capacity and g_4 barrels per day of gasoline capacity. This implies $f_4(o_4, g_4)= c_4(o_4, g_4)$. For t=1, 2, 3, we can determine $f_t(o_t, g_t)$ by noting that if we build a refinery at site t that can refine x_t barrels of oil per day and y_t barrels of gasoline per day, then we incur a cost of $c_t(x_t, y_t)$ at site t. Then we will need to build a total refinery capacity of o_t-x_t and a gas refinery capacity of g_t-y_t at sites t+1, t+2,...,4. By principle of optimality, the cost of doing this will be $f_{t+1}(o_t$-x_t, g_t-$y_t)$. Since $0 \le x_t \le o_t$ and $0 \le y_t \le g_t$ must hold, we obtain the following recursion:

$$f_t(o_t, g_t) = \min\{c_t(o_t, g_t) + f_{t+1}(o_t - x_t, g_t - y_t)\} \qquad (17)$$

$0 \le x_t \le o_t$ and $0 \le y_t \le g_t$

Example 6.12. Cash Registers *(Winston 18.6., p. 1000 q.3)*

Assume that during minute t, the following sequence of events occurs: (1) at the beginning of the minute, x_t customers arrive at the cash register; (2) the store manager decides how many cash registers should be operated during the current minute; (3) if s cash registers are operated and i customers are present (including the current minute's arrivals), $g(s,i)$ customers complete service; and (4) the next minute begins.

A cost of 10 cent is assessed for each minute a customer spends waiting to check out. Assume that it costs c(s) cents to operate s cash registers for 1 minute. Formulate a dynamic programming recursion that minimizes the sum of holding and service costs during the next 60 minutes. Assume that before the first minute's arrivals, no customers are present and that holding cost is assessed at the end of each minute.

6.7.2 Solving Recursions

A dynamin programming formulation can be solved by using any programming language such as python, Matlab, C++, etc. The parameters and the functions defined in the problem should be provided for solution.

Example 6.13. Solution of the Fishery Example

Write a Python code to solve the Fisher example for the given functions and parameters:

$$r(x) = 40x^{0.5} \qquad c(x,b) = \frac{120x}{b} \qquad \beta = 0.95$$

Assume that x is determined in thousands and next year's bass is calculated by the function $Round(1.2*(b-x))$.

Answer

```
#DEFINE RECURSION

def bassprofit(t,b,T,beta):
    if t == T + 1 :
#        print(t,b,'T+1 degeri =', 0)
         return 0
    if fonk[t,b] is not None:
#        print(t,b,'previously found =', fonk[t,b])
         return fonk[t,b]
    else:
       fonk[t,b] = -1
       for x in range(b+1):
           deger = (revenue(x)-cost(x,b)+ beta*bassprofit(t+1,round(1.2*(b-x)),T,beta))
           if deger > fonk[t,b]:
               fonk[t,b]= deger
               decision[t,b]="x="+str(x)+" t="+str(t+1)+" b=" + str(round(1.2*(b-x)))
               decisionx[t,b] = x
               nextb[t,b] = round(1.2*(b-x))
       print('f(',t,b,') =', fonk[t,b], '-- Decision = ', decision[t,b])
       return fonk[t,b]

#DEFINE Revenue and Cost functions
def revenue(x):
    return 40*(x**0.5)

def cost(x,b):
    if b == 0 or x == 0:
        return 0
    else:
        return x*120/b

#DEFINE parameters
t = 1
b = 20
T = 10
beta = 0.95

#DEFINE LISTs to store data
TT = list(tp for tp in range(T+1) )
B = list(bp for bp in range(round(b*(1.2)**T)+2))
```

```
fonk = {(tp,bp):None for tp in TT for bp in B}
decision ={(tp,bp):None for tp in TT for bp in B}
decisionx ={(tp,bp):None for tp in TT for bp in B}
nextb ={(tp,bp):None for tp in TT for bp in B}

# RUN
maxprofit = bassprofit(t,b,T,beta)

#PRINT SOLUTION
print('------------------------------------')
print('---------------SOLUTION-------------')
print('------------------------------------')

print('Maximum Profit is : ',maxprofit)
print('------------------------------------')

bb = b
for t in range(T):
    print('year ',t+1,': Initial bass:', bb, '// Catch',decisionx[t+1,bb],' bass')
    bb = nextb[t+1,bb]
```

6.7.3 Nonadditive Recursions

Example 6.14. Multiplicative Recursion

Solve the following mathematical program using dynamic programming.

$$Max\ g_1(x_1)g_2(x_2)g_3(x_3)$$

Subject to $x_1 + x_2 + x_3 = 4$

All variables are non-negative and integer, the functions are defined as follows:

$g_1(x) = x, \quad g_2(x) = 2 - x, \quad g_3(x) = x^2$

Answer

To solve the given model using dynamic programming the following recursion can be used:

$$f_t(i) = max_x[g_t(x)f_{t+1}(i-x)]$$

Stage 3 calculations:

$f_3(0) = 0 \quad f_3(1) = 1 \quad f_3(2) = 4 \quad f_3(3) = 9 \quad f_3(4) = 16$

Stage 2 calculations:

$f_2(4) = 32 \quad f_2(3) = 18 \quad f_2(2) = 8 \quad f_2(1) = 2 \quad f_2(0) = 0$

Stage 2 calculation:

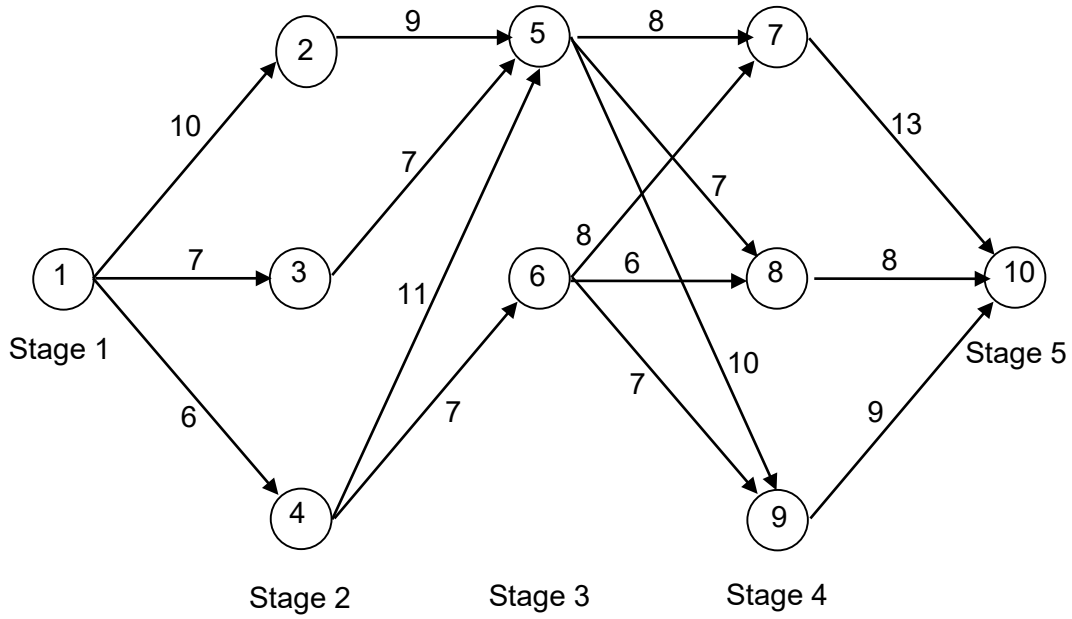
$f_1(4) = 18$

Solution: $x_1 = 1, x_2 = 0, x_3 = 3$

Example 6.15. Minimax Shortest Route *(Winston 18.6, p.997)*

Joe Cougar needs to drive from city 1 to city 10. He is no longer interested in minimizing the length of his trip, but he is interested in minimizing the maximum altitude above sea level that he will encounter during his drive. To get from city 1 to city 10, he must follow a path in Figure 1. The length c_{ij} of the arc connecting city *i* and city *j* represents the maximum altitude

(in thousands of feet above the sea level) encountered when driving from city *i* to city *j*. Use DP to determine how Joe should proceed from city 1 to city 10.



Answer

To solve this problem by DP, note that for a trip that begins in a city *i* and goes through stages *t, t+1,...,5*, the maximum altitude that Joe encounters will be the maximum of the following two quantities: (1) the maximum altitude encountered on stages *t+1, t+2,...,5* or (2) the altitude encountered when traversing the arc that begins in stage *t*. Of course, if we are in a stage 4 state, quantity 1 does not exist.

After defining $f_t(i)$ as the smallest maximum altitude that Joe can encounter in a trip from a city *i* in stage *t* to city 10, this reasoning leads us to the following recursion:

$$f_4(i)=c_{i,10} \tag{18}$$

$$f_t(i)=\min_j\{\max(c_{ij},f_{t+1}(j))\} \qquad (t=1,\ 2,\ 3)$$

where *j* may be any city such that there is an arc connecting city *i* and city *j*.

We first compute $f_4(7)$, $f_4(8)$, and $f_4(9)$ and than use (18) to work backward until $f_1(1)$ has been computed. We obtain the following results:

$$f_4(7)=13\ *$$
$$f_4(8)=8\ *$$
$$f_4(9)=9\ *$$

$$f_3(5)=\min\begin{cases}\max(c_{57},f_4(7))=13\\ \max(c_{58},f_4(8))=8\ *\\ \max(c_{59},f_4(9))=10\end{cases}$$

$$f_3(6)=\min\begin{cases}\max\left(c_{67},f_4(7)\right)=13\\ \max\left(c_{68},f_4(8)\right)=8^*\\ \max\left(c_{69},f_4(9)\right)=9\end{cases}$$

$$f_2(2)=\max\left(c_{25},f_3(5)\right)=8^*$$
$$f_2(3)=\max\left(c_{35},f_3(5)\right)=9^*$$

$$f_2(4)=\min\begin{cases}\max\left(c_{45},f_3(5)\right)=11\\ \max\left(c_{46},f_3(6)\right)=8^*\end{cases}$$

$$f_1(1)=\min\begin{cases}\max\left(c_{12},f_2(2)\right)=10\\ \max\left(c_{13},f_2(3)\right)=8^*\\ \max\left(c_{14},f_2(4)\right)=8^*\end{cases}$$

To determine the optimal strategy, note that Joe can begin by going from city 1 to city 3 or from city 1 to city 4. Suppose Joe begins by traveling to city 3. Then he should choose the arc attaining $f_2(3)$, which means he should next travel to city 5. Then Joe must choose the arc that attains $f_3(5)$, driving next to city 8. Then, of course, he must drive to city 10. Thus, the path 1-3-5-8-10 is optimal, and Joe will encounter a maximum altitude equal to $f_1(1)$=8,000 ft. You can verify that the path 1-4-6-8-10 is also optimal.

6.8 Wagner-Within Algorithm and Silver-Meal Heuristic

The inventory example is a special case of the *dynamic lot-size model.*

6.8.1 Description of Dynamic Lot-Size Model

1. Demand d_t during period t (t=1, 2, …, T) is known at the beginning of period 1.
2. Demand for period t must be met on time from inventory or from period t production. The cost $c(x)$ of producing x units during any period is given by $c(0)=0$, and for $x>0$, $c(x)=K+cx$, where K is a fixed cost for setting up production during a period, and c is the variable per-unit cost of production.
3. At the end of period t, the inventory level i_t is observed, and a holding cost hi_t is incurred. We let i_0 denote the inventory level before period 1 production occurs.
4. The goal is to determine a production level x_i for each period t that minimizes the total cost of meeting (on time) the demands for periods 1, 2, …, T.
5. There is a limit c_t placed on period t's ending inventory.
6. There is a limit r_t placed on period t's production.

Example 6.16. Dynamic Lot-Size Model *(Winston 18.7, p.1002)*

We now determine an optimal production schedule for a five-period dynamic lot-size model with K = \$30, c = \$5, h= \$1, d_1 = 40, d_2 = 60, d_3 = 10, d_4 = 20 and d_5 = 25. We assume that the initial inventory level is zero.

6.8.2 Discussion of the Wagner-Whitin Algorithm

If the DP approach were used to find an optimal production policy, we would have to consider the possibility of producing any amount between 0 and $d_1+d_2+d_3+d_4+d_5$ = 155 units during period 1. Thus, it would be possible for the period 2 state (period 2's entering inventory) to be 0, 1, ..., 155 – d_1 = 115, and we would have to determine $f_2(0)$, $f_2(1)$, ..., $f_2(115)$. Using the DP approach to find an optimal production schedule would therefore require a great deal of computational effort. Fortunately, however, Wagner and Whitin (1958) have developed a method that greatly simplifies the computation of optimal production schedules for dynamic lot-size models. Lemmas 1 and 2 are necessary for the development of the Wagner-Whitin algorithm.

LEMMA 1

Suppose it is optimal to produce a positive quantity during a period t. Then for some j = 0, 1, ..., $T – t$, the amount produced during period t must be such that after period t's production, a quantity $d_t + d_{t+1} + ... + d_{t+j}$ will be in stock. In other words, if production occurs during period t, we must (for some j) produce an amount that exactly suffices to meet the demands for periods $t, t+1, ..., t+j$.

LEMMA 2

If it is optimal to produce anything during period t, then $i_{t-1} < d_t$. In other words, production cannot occur during period t unless there is insufficient stock to meet period t demand.

Please see Winston (2004) Section 18.4 for proof of lemmas.

Wagner and Whitin Algorithm Recursion

We assume that the initial inventory level is zero. Let f_t is the minimum cost incurred during periods t, t+1,…, T given that at the beginning of period t, the inventory level is zero. Notice that if the inventory level at the beginning of period t is zero we have to produce at least as much as the demand of period t, which means that we make production on period t. Then f_t (t=1,…,T) is calculated as

$$f_t = \min_{j=0,1,2,...T-t} \left(c_{tj} + f_{t+j+1} \right) \qquad (20)$$

where $c_{tj} = K + c(d_t + d_{t+1} + ... + d_{t+j}) + h(d_{t+1} + 2*d_{t+2} + ... + j*d_{t+j})$.

To find an optimal production schedule by the Wagner–Whitin algorithm, begin by $f_{T+1} = 0$, and use the given recursion to compute $f_T, f_{T-1}, \ldots, f_1$. Once f_1 has been found, an optimal production schedule may be easily obtained.

Example 6.17. Dynamic Lot-Size Model (continued)

Answer

To illustrate the Wagner-Whitin algorithm, we find an optimal production schedule. The computations follow.

$f_6 = 0$

$f_5 = 30 + 5(25) + f_6 = 155$ (Produce for period 5)

If we begin period 5 with zero inventory, we have to produce for period 5, and we can only produce for meeting the demand of period 5.

$$f_4 = \min\begin{cases} 30 + 5(20) + f_5 = 285 & \text{(Produce for period 4)} \\ 30 + 5(20 + 25) + 1(25) + f_6 = 280^* & \text{(Produce for periods 4,5)} \end{cases}$$

If we begin period 4 with zero inventory, we have to produce at least for period 4. We may produce for period 4 only or for periods 4 and 5. As the result of the above given calculations, if we produce on period 4, it costs less to produce for periods 4 and 5 to meet the demands of periods 4 and 5.

$$f_3 = \min\begin{cases} 30 + 5(10) + f_4 = 360 & \text{(Produce for period 3)} \\ 30 + 5(10 + 20) + 1(20) + f_5 = 355^* & \text{(Produce for periods 3, 4)} \\ 30 + 5(10 + 20 + 25) + 1(20 + 2(25)) + f_6 = 375 & \text{(Produce for periods 3, 4, 5)} \end{cases}$$

If we begin period 3 with zero inventory, we have to produce at least for period 3. However, we may produce for periods 3 and 4 or for periods 3,4, and 5. Calculations show that it costs less to produce for periods 3 and 4 if we produce on period 3.

$$f_2 = \min\begin{cases} 30 + 5(60) + f_3 = 685 & \text{(Produce for period 2)} \\ 30 + 5(60 + 10) + 1(10) + f_4 = 670^* & \text{(Produce for period 2,3)} \\ 30 + 5(60 + 10 + 20) + 1(10 + 2(20)) + f_5 = 685 & \text{(Produce for period 2,3,4)} \\ 30 + 5(60 + 10 + 20 + 25) + 1(10 + 2(20) + 3(25)) + f_6 = 730 & \text{(Produce for period 2,3,4,5)} \end{cases}$$

If we begin period 2 with zero inventory, we should produce enough during period 2 to meet the demand for period 2. We may also produce for period 2 and 3, or for periods 2,3, and 4 or for periods 2, 3, 4, and 5. Results show that it costs less to produce for periods 2 and 3 if we produce on period 2.

$$f_1 = \min\begin{cases} 30+5(40)+f_2 = 900^* & \text{(Produce for period 1)} \\ 30+5(40+60)+1(60)+f_3 = 945 & \text{(Produce for period 1,2)} \\ 30+5(40+60+10)+1(60+2(10))+f_4 = 940 & \text{(Produce for period 1,2,3)} \\ 30+5(40+60+10+20)+1(60+2(10)+3(20))+f_5 = 975 & \text{(Produce for period 1,2,3,4)} \\ 30+5(40+60+10+20+25)+1(60+2(10)+3(20)+4(25)) & \\ \qquad +f_6 = 1045 & \text{(Produce for period 1,2,3,4,5)} \end{cases}$$

If we begin period 1 with zero inventory, there are 5 options as given above. The optimal option is to produce only for period 1.

Optimal production schedule

We go forward to starting from period 1 (f_1) to find the optimal schedule. It is optimal to produce $x_1 = d_1 = 40$ units during period 1; then we begin period 2 with zero inventory. Since f_2 is attained by producing demand of period 2 and 3, we should produce $x_2 = d_2 + d_3 = 70$ units during period 2; then we enter period 4 with zero inventory (notice that since we meet the demand of period 3 by the production of period 2, we skip period 3). Since f_4 is attained by meeting the demands for periods 4 and 5, we produce $x_4 = d_4 + d_5 = 45$ units during period 4. Since we don't produce on periods 3 and 5, $x_3 = x_5 = 0$. The optimal production schedule will incur a total cost of f_1 = \$900. Any optimal production schedule must produce exactly $d_1+d_2+d_3+d_4+d_5 = 155$ units,

Incurring variable production costs of 5(155)=\$775. Thus, in computing the optimal production schedule, we may always ignore the variable production costs. This substantially simplifies the calculations.

6.8.3 The Silver-Meal Heuristic

The Silver-Meal (S-M) heuristic involves less work than the Wagner-Whitin algorithm and can be used to find a near-optimal production schedule. The S-M heuristic is based on the fact that our goal is to minimize the average cost per period (for the reasons stated, variable production costs may be ignored). Suppose we are at the beginning of period 1 and are trying to determine how many periods of demand should be satisfied by period 1's production. During period 1, if we produce an amount sufficient to meet demand for the next t periods, then a cost of $TC(t)=K+HC(t)$ will be incurred (ignoring variable production costs). Here $HC(t)$ is the holding cost incurred during the next t periods (including the current period) if production during the current period is sufficient to meet demand for the next t periods.

Let $AC(t) = \frac{TC(t)}{t}$ be the average per-period cost incurred during the next t periods. Since $\frac{1}{t}$ is a decreasing convex function of t, as t increases $\frac{K}{t}$ decreases at a decreasing rate. In most cases, $\frac{HC(t)}{t}$ tends to be an increasing function of t. Thus, in most situations, an integer

t^* can be found such that for $t<t^*$, $AC(t+1) \leq AC(t)$ and $AC(t^*+1) \geq AC(t^*)$. The S-M heuristic recommends that period 1's production be sufficient to meet the demands for periods 1, 2, …, t^* (if no t^* exists, period 1 production should satisfy the demand for periods 1, 2, …, T). Since t^* is a local (and perhaps global) minimum for $AC(t)$, it seems reasonable that producing $d_1 + d_2 + \ldots + d_{t^*}$ units during period 1 will come close to minimizing the average per-period cost incurred during periods 1, 2, …, t^*. Next we apply the S-M heuristic while considering period t^*+1 as the initial period. We find that during period t^*+1, the demand for the next t_1^* periods should be produced. Continue in this fashion until the demand for period T has been produced.

To illustrate, we apply the S-M heuristic to the lot-size question. We have

$TC(1) = 30$ $\qquad$ $AC(1) = 30 / 1 = 30$

$TC(2) = 30 + 60 = 90$ $\qquad$ $AC(2) = 90 / 2 = 45$

Since $AC(2) \geq AC(1)$, $t^*=1$, and the S-M heuristic dictates that we produce $d_1 = 40$ units during period 1. Then for period 2;

$TC(1) = 30$ $\qquad$ $AC(1) = 30 / 1 = 30$

$TC(2) = 30 + 10 = 40$ $\qquad$ $AC(2) = 40 / 2 = 20$

$TC(3) = 30 + 10 + 2(20) = 80$ $\qquad$ $AC(3) = 80 / 3 = 26{,}67$

Since $AC(3) \geq AC(2)$, the S-M heuristic recommends producing $d_2 + d_3 = 60 + 10 = 70$ units during period 2. Then for period 4;

$TC(1) = 30$ $\qquad$ $AC(1) = 30 / 1 = 30$

$TC(2) = 30 + 25 = 55$ $\qquad$ $AC(2) = 55 / 2 = 27{,}5$

Since we cannot produce for more than 2 periods on period 4 (i.e., there is no period 6) we stop. Since $AC(2) \leq AC(1)$, period 4 production should meet the demand for the next two periods (periods 4 and 5). During period 4, we should produce $d_4 + d_5 = 45$ units. Notice that the schedule found by S-M heuristic is exactly the same with the one of Wagner- Whitin algorithm.

For many dynamic lot-size problems, the S-M heuristic yields an optimal production schedule. In extensive testing, the S-M heuristic usually yielded a production schedule costing less than 1% above the optimal policy obtained by the Wagner-Whitin algorithm (see Peterson and Silver (1998)).

7. PROBABILISTIC DYNAMIC PROGRAMMING

In deterministic dynamic programming, a specification of the current state and current decision was enough to tell us with certainty the new state and the costs during the current stage. In many practical problems, these factors may not be known with certainty, even if the current state and decision are known.

In this part, we explain how to use dynamic programming to solve problems in which the current period's cost or the next period's state are random. These problems are called probabilistic dynamic programming problems (or PDPs). In a PDP, the decision maker's goal is usually to minimize expected (or expected discounted) cost incurred or to maximize expected (or expected discounted) reward earned over a given time horizon.

7.1 Current Stage Costs are Uncertain and Next Period's State is Certain

For this kind of problems, the next period's state is known with certainty, but the reward earned during the current stage is not known with certainty (given the current state and decision).

Example 7.1. Milk Distribution *(Winston 19.1, p. 1016)*

For a price of $1/gallon, the Safeco Supermarket chain has purchased 6 gallons of milk from a local dairy. Each gallon of milk is sold in the chain's three stores for $2/gallon. The dairy must buy back for 50¢/gallon any milk that is left at the end of the day. Unfortunately for Safeco, demand for each of the chain's three stores is uncertain. Past data indicate that the daily demand at each store is as shown in the following Table. Safeco wants to allocate the 6 gallons of milk to the three stores so as to maximize the expected net daily profit (revenues less costs) earned from milk.

Probability Distributions for Daily Milk Demand

	Daily Demand (gallons)	Probability
Store 1	1	.60
	2	0
	3	.40
Store 2	1	.50
	2	.10
	3	.40
Store 3	1	.40
	2	.30
	3	.30

Use dynamic programming to determine how Safeco should allocate the 6 gallons of milk among the three stores.

Answer

Define;

$r_t(g_t)$ = expected revenue earned from g_t gallons assigned to store t

$f_t(x)$ = maximum expected revenue earned from x gallons assigned to stores t, t+1,…,3.

For t = 3; $f_3(x) = r_3(x)$

For t = 1,2; $f_t(x) = \max_{g_t}\{r_t(g_t) + f_{t+1}(x - g_t)\}$

where g_t must be a member of {0,1,…,x}.

We begin by computing the $r_t(g_t)$'s:

$r_3(0)$ = \$0	$r_2(0)$ = \$0	$r_1(0)$ = \$0
$r_3(1)$ = \$2.00	$r_2(1)$ = \$2.00	$r_1(1)$ = \$2.00
$r_3(2)$ = \$3.40	$r_2(2)$ = \$3.25	$r_1(2)$ = \$3.10
$r_3(3)$ = \$4.35	$r_2(3)$ = \$4.35	$r_1(3)$ = \$4.20

We determine an optimal allocation of milk to stores. First compute $f_3(x)$:

$f_3(0) = r_3(0) = 0$	$g_3(0) = 0$
$f_3(1) = r_3(1) = 2.00$	$g_3(1) = 1$
$f_3(2) = r_3(2) = 3.40$	$g_3(2) = 2$
$f_3(3) = r_3(3) = 4.35$	$g_3(3) = 3$

Work backward and obtain $f_2(x)$:

$f_2(0) = r_2(0) + f_2(0-0) = 0 \qquad g_2(0) = 0$

$$f_2(1) = \max\begin{cases} r_2(0) + f_3(1-0) = 2 * \\ r_2(1) + f_3(1-1) = 2 * \end{cases} \qquad g_2(1) = 0 \ \ or\ 1$$

$$f_2(2) = \max\begin{cases} r_2(0) + f_3(2-0) = 0 + 3.4 = 3.4 \\ r_2(1) + f_3(2-1) = 2 + 2 = 4 * \\ r_2(2) + f_3(2-2) = 3.25 + 0 = 3.25 \end{cases} \qquad g_2(2) = 1$$

$$f_2(3) = \max\begin{cases} r_2(0) + f_3(3-0) = 0 + 4.35 = 4.35 \\ r_2(1) + f_3(3-1) = 2 + 3.4 = 5.4 * \\ r_2(2) + f_3(3-2) = 3.25 + 2 = 5.25 \\ r_2(3) + f_3(3-3) = 4.35 + 0 = 4.35 \end{cases} \qquad g_2(3) = 1$$

$$f_2(4) = \max\begin{cases} r_2(1) + f_3(4-1) = 2 + 4.35 = 6.35 \\ r_2(2) + f_3(4-2) = 3.25 + 3.4 = 6.65 * \\ r_2(3) + f_3(4-3) = 4.35 + 2 = 6.35 \end{cases} \qquad g_2(4) = 2$$

$$f_2(5) = \max\begin{cases} r_2(2) + f_3(5-2) = 3.25 + 4.35 = 7.6 \\ r_2(3) + f_3(5-3) = 4.35 + 3.4 = 7.75 * \end{cases} \qquad g_2(5) = 3$$

$f_2(6) = r_2(3) + f_3(6-3) = 4.35 + 4.35 = 8.7 \qquad g_2(6) = 3$

Finally,

$$f_1(6) = \max \begin{cases} r_1(0) + f_2(6-0) = 0 + 8.7 = 8.7 \\ r_1(1) + f_2(6-1) = 2 + 7.75 = 9.75 * \\ r_1(2) + f_2(6-2) = 3.1 + 6.65 = 9.75 * \\ r_1(3) + f_2(6-3) = 4.2 + 5.4 = 9.6 \end{cases} \qquad g_1(6) = 1 \text{ or } 2$$

Thus, we can either assign 1 or 2 gallons to store 1. Suppose we arbitrarily choose to assign 1 gallon to store 1. Then we have 6 – 1 = 5 gallons for stores 2 and 3. Since $f_2(5)$ is attained by $g_2(5) = 3$, we assign 3 gallons to store 2. Then 5 – 3 = 2 gallons are available for store 3. Since $g_3(2) = 2$, we assign 2 gallons to store 3. Note that although this policy obtains the maximum expected revenue, $f_1(6) = \$9.75$, the total revenue actually received on a given day may be more or less than \$9.75. For example, if demand at each store were 1 gallon, total revenue would be 3(2.00) + 3(0.50) = \$7.50, whereas if demand at each store were 3 gallons, all the milk would be sold at \$2/gallon, and the total revenue would be 6(2.00) = \$12.00.

7.2 A Probabilistic Inventory Model

When the demand is uncertain in an inventory model it can be solved using a PDP for which the state during the next period is uncertain (given the current state and current decision).

Example 7.2. Three-Period Production Policy *(Winston 19.2, p. 1019)*

At the beginning of each period, a firm must determine how many units should be produced during the current period. During a period in which x units are produced, a production cost $c(x)$ is incurred, where $c(0)=0$, and for $x > 0$, $c(x) = 3 + 2x$. Production during each period is limited to at most 4 units. After production occurs, the period's random demand is observed. Each period's demand is equally likely to be 1 or 2 units. After meeting the current period's demand out of current production and inventory, the firm's end-of-period inventory is evaluated, and a holding cost of \$1 per unit is assessed. Because of limited capacity, the inventory at the end of each period cannot exceed 3 units. It is required that all demand be met on time. Any inventory on hand at the end of period 3 can be sold at \$2 per unit. At the beginning of period 1, the firm has 1 unit of inventory. Use dynamic programming to determine a production policy that minimizes the expected net cost incurred during the three periods.

Answer

Define $f_t(i)$ to be the minimum expected net cost incurred during periods t, t+1,...,3 when the inventory at the beginning of period t is i units. Then

$$f_3(i) = \min_x \left\{ c(x) + \left(\frac{1}{2}\right)(i + x - 1) + \left(\frac{1}{2}\right)(i + x - 2) - \left(\frac{1}{2}\right)2(i + x - 1) - \left(\frac{1}{2}\right)2(i + x - 2) \right\}$$

where x must be a member of {0, 1, 2, 3, 4} and x must satisfy $(2 - i) \le x \le (4 - i)$.

The given Equation follows, because if x units are produced during period 3, the net cost during period 3 is (expected production cost) + (expected holding cost) - (expected salvage value). If x units are produced, the expected production cost is $c(x)$, and there is a 1/2 chance that the period 3 holding cost will be $i + x - 1$ and a 1/2 chance that it will be $i + x - 2$. Hence, the period 3 expected holding cost will be $\left(\frac{1}{2}\right)(i + x - 1) + \left(\frac{1}{2}\right)(i + x - 2) = i + x - \frac{3}{2}$. Similar reasoning shows that the expected salvage value (a negative cost) at the end of period 3 will be $\left(\frac{1}{2}\right)2(i + x - 1) + \left(\frac{1}{2}\right)2(i + x - 2) = 2i + 2x - 3$. To ensure that period 3 demand is met, we must have $i + x \geq 2$, or $x \geq 2 - i$. Similarly, to ensure that ending period three inventory does not exceed 3 units, we must have $i + x - 1 \leq 3$, or $x \leq 4 - i$.

For $t = 1, 2$, we can derive the recursive relation for $f_t(i)$ by noting that for any month t production level x, the expected costs incurred during periods t, $t + 1$,..., 3 are the sum of the expected costs incurred during period t and the expected costs incurred during periods t+1, t+2,...,3. As before, if x units are produced during month t, the expected cost during month t will be $c(x) + \left(\frac{1}{2}\right)(i + x - 1) + \left(\frac{1}{2}\right)(i + x - 2)$. (Note that during periods 1 and 2, no salvage value is received.) If x units are produced during month t, the expected cost during periods t +1, t +2,…, 3 is computed as follows. Half of the time, the demand during period t will be 1 unit, and the inventory at the beginning of period $t + 1$ will be $i + x - 1$. In this situation, the expected costs incurred during periods t+1, t+2,...,3 (assuming we act optimally during these periods) is $f_{t+1}(i+x-1)$. Similarly, there is a 1/2 chance that the inventory at the beginning of period t+1 will be $i+x-2$. In this case, the expected cost incurred during periods t+1, t+2,...,3 will be $f_{t+1}(i+x-2)$. In summary, the expected cost during periods t+1, t+2,...,3 will be $\left(\frac{1}{2}\right)f_{t+1}(i + x - 1) + \left(\frac{1}{2}\right)f_{t+1}(i + x - 2)$. With this in mind, we may write for $t = 1, 2$,

$$f_t(i) = \min_x \left\{ c(x) + \left(\frac{1}{2}\right)(i + x - 1) + \left(\frac{1}{2}\right)(i + x - 2) + \left(\frac{1}{2}\right)f_{t+1}(i + x - 1) - \left(\frac{1}{2}\right)f_{t+1}(i + x - 2) \right\}$$

where x must be a member of {0, 1, 2, 3, 4} and x must satisfy $(2 - i) \leq x \leq (4 - i)$.

We define $x_t(i)$ to be a period t production level attaining the minimum in the previous equation for $f_t(i)$. We now work backward until $f_1(1)$ is determined. The relevant computations are summarized in the following Tables. Since each period's ending inventory must be nonnegative and cannot exceed 3 units, the state during each period must be 0, 1, 2, or 3.

Computations for $f_3(i)$

i	x	$c(x)$	Expected holding cost $i + x - \frac{3}{2}$	Expected Salvage Value $2i + 2x - 3$	Total Expected cost	$f_3(i)$ $x_3(i)$

3	0	0	1.5	3	-1.5*	$f_3(3) = -1.5$
3	1	5	2.5	5	2.5	$x_3(3) = 0$
2	0	0	0.5	1	-0.5*	$f_3(2) = -0.5$
2	1	5	1.5	3	3.5	$x_3(2) = 0$
2	2	7	2.5	5	4.5	
1	1	5	0.5	1	4.5*	$f_3(1) = 4.5$
1	2	7	1.5	3	5.5	$x_3(1) = 1$
1	3	9	2.5	5	6.5	
0	2	7	0.5	1	6.5*	$f_3(0) = 6.5$
0	3	9	1.5	3	7.5	$x_3(0) = 2$
0	4	11	2.5	5	8.5	

Computations for $f_2(i)$

i	x	$c(x)$	Expected holding cost $i + x - \frac{3}{2}$	Expected Future Cost $\left(\frac{1}{2}\right) f_3(i + x - 1) + \left(\frac{1}{2}\right) f_3(i + x - 2).$	Total Expected cost Periods 2,3	$f_2(i)$ $x_2(i)$
3	0	0	1.5	2	3.5*	$f_2(3) = 3.5$
3	1	5	2.5	-1	6.5	$x_2(3) = 0$
2	0	0	0.5	5.5	6*	$f_2(2) = 6$
2	1	5	1.5	2	8.5	$x_2(2) = 0$
2	2	7	2.5	-1	8.5	
1	1	5	0.5	5.5	11	$f_2(1) = 10.5$
1	2	7	1.5	2	10.5*	$x_2(1) = 2$ or 3
1	3	9	2.5	-1	10.5*	
0	2	7	0.5	5.5	13	$f_2(0) = 12.5$
0	3	9	1.5	2	12.5*	$x_2(0) = 3$ or 4
0	4	11	2.5	-1	12.5*	

Computations for $f_1(1)$

i	x	$c(x)$	Expected holding cost $i + x - \frac{3}{2}$	Expected Future Cost $\left(\frac{1}{2}\right) f_2(i + x - 1) + \left(\frac{1}{2}\right) f_2(i + x - 2).$	Total Expected cost Periods 1,3	$f_1(i)$ $x_1(i)$
1	1	5	0.5	11.5	17	$f_1(1) = 16.25$
1	2	7	1.5	8.25	16.75	$x_1(1) = 3$
1	3	9	2.5	4.75	16.25*	

We begin by producing $x_1(1) = 3$ units during period 1. We cannot, however, determine period 2's production level until period 1's demand is observed. Also, period 3's production level cannot be determined until period 2's demand is observed. To illustrate the idea, we determine the optimal production schedule if period 1 and period 2 demands are both 2 units. Since $x_1(1)= 3$, 3 units will be produced during period 1. Then period 2 will begin with an inventory of 1 + 3 – 2 = 2 units, so $x_2(2) = 0$ units should be produced. After period 2's demand of 2 units is met, period 3 will begin with 2 – 2 = 0 units on hand. Thus, $x_3(0) = 2$ units will be produced during period 3.

In contrast, suppose that period 1 and period 2 demands are both 1 unit. As before, $x_1(1) = 3$ units will be produced during period 1. Then period 2 will begin with 1 + 3 - 1 = 3 units, and $x_2(3) = 0$ units will be produced during period 2. Then period 3 will begin with 3 - 1 = 2 units on hand, and $x_3(2) = 0$ units will be produced during period 3. Note that the optimal production policy has adapted to the low demand by reducing period 3 production.

This example illustrates an important aspect of dynamic programming solutions for problems in which future states are not known with certainty at the beginning of the problem: *If a* random *factor (such as random demand) influences transitions from the period t state to the period t + 1 state, t*he *optimal action for period t cannot be determined until period t's state is known.*

7.3 How to Maximize the Probability of a Favorable Event Occurring

There are many occasions on which the decision maker's goal is to maximize the probability of a favorable event occurring. For instance, a company may want to maximize its probability of reaching a specified level of annual profits. To solve such a problem, we assign a reward of 1 if the favorable event occurs and a reward of 0 if it does not occur. Then the maximization of expected reward will be equivalent to maximizing the probability that the favorable event will occur. Also, the maximum expected reward will equal the maximum probability of the favorable event occurring.

Example 7.3. Sales Allocation *(Winston 18.6, p.998)*

Glueco is planning to introduce a new product in three different regions. Current estimates are that the product will sell well in each region with respective probabilities. .6, .5, and .3. The firm has available two top sales representatives that it can send to any of the three regions. The estimated probabilities that the product will sell well in each region when 0, 1, or 2 additional sales reps are sent to a region are given in Table 16. If Glueco wants to maximize the probability that its new product will sell well in all three regions, than where

should it assign sales representatives? You may assume that sales in the three regions are independent.

Number of Additional Sales Representatives	Probability of Selling Well		
	Region 1	Region 2	Region 3
0	0.6	0.5	0.3
1	0.8	0.7	0.55
2	0.85	0.85	0.7

Answers

If Glueco had just one region to worry about and wanted to maximize the probability that the new product would sell in that region, then the proper strategy would be clear: Assign both sales reps to the region. We could then work backward and solve a problem in which Glueco's goal is to maximize the probability that the product will sell in two regions. Finally, we could work backward and solve a problem with three regions. We define $f_t(s)$ as the probability that the new product will sell in regions *t, t+1,...*3 if *s* sales reps are optimally assigned to these regions. Then

$f_3(2)=0.7$ (2 reps allocated to region 3.)

$f_3(1)=0.55$ (1 rep allocated to region 3.)

$f_3(0)=0.3$ (0 rep allocated to region 3.)

Also, $f_1(2)$ will be the maximum probability that the product will sell well in all three regions. To develop a recursion for $f_2(-)$ and $f_1(-)$, we define p_{tx} to be the probability that the new product sells well in region *t* if *x* sales reps are assigned to region *t*. For example, $p_{21}= 0.7$. For *t*=1 and *t*=2, we then write

$$f_t(s)=\max_x\{p_{tx}f_{t+1}(s-x)\} \qquad (19)$$

where *x* must be a member of {0,1,...,s}. To justify (19), observe that if *s* sales reps are available for regions t, t+1,...3 and *x* sales reps are assigned to region *t*, then

p_{tx}= probability that products sells in region *t*

$f_{t+1}(s-x)$= probability that product sells well in regions t+1,..., 3

Note that the sales in each region are independent. This implies that if *x* sales reps are assigned to region *t*, then the probability that the new product sells well in regions *t*, t, t+1,..., 3 is $p_{tx}f_{t+1}(s-x)$. We want to maximize this probability, so we obtain (19). Applying (19) yields the following results:

$$f_2(2)=\max\begin{cases}0{,}5f_3(2-0) &= 0{,}35\\ 0{,}7f_3(2-1) &= 0{,}385^*\\ 0{,}85f_3(2-2) &= 0{,}225\end{cases}$$

Thus $f_2(2)$=0,385 and 1 sales rep should be assigned to region 2.

$$f_2(1)=\max\begin{cases}0{,}5f_3(1-0)=0{,}275\\0{,}5f_3(1-1)=0{,}21\end{cases}$$

Thus $f_2(2)$=0,275 and no sales reps should be assigned to region 2.

$$f_2(0)=\{0{,}5f_3(0-0)\}=0{,}15$$

Finally, we are back to the original problem, which is to find $f_1(2)$. Equation (19) yields

$$f_1(2)=\max\begin{cases}(0{,}6f_2(2-0))=0{,}231^*\\(0{,}8f_2(2-1))=0{,}220\\(0{,}85f_2(2-2))=0{,}1275\end{cases}$$

Thus $f_1(2)$=.231, and no sales reps should be assigned to region 1. Then Glueco needs to attain $f_2(2\text{-}0)$, which requires that 1 sales rep be assigned to region 2. Glueco must next attain $f_3(2\text{-}1)$, which requires that 1 sales rep be assigned to region 3. In summary' Glueco can obtain a .231 probability of the new product selling well in all three regions by assigning 1 sales rep to region 2 and 1 sales rep to region 3

Example 7.4. Tennis Serves *(Winston 19.3, p. 1026)*

Martina McEnroe has two types of serves: a hard serve (*H*) and a soft serve (*S*). The probability that Martina's hard serve will land in bounds is p_H, and the probability that her soft serve will land in bounds is p_S. If Martina's hard serve lands in bounds, there is a probability w_H that Martina will win the point. If Martina's soft serve lands in bounds, there is a probability w_S that Martina will win the point. We assume that $p_H < p_S$ and $w_H > w_S$. Martina's goal is to maximize the probability of winning a point on which she serves. Use dynamic programming to help Martina select an optimal serving strategy. Remember that if both serves are out of bounds, Martina loses the point.

Example 7.5. Chess game *(Winston 19.3, p.1028)*

Vladimir Ulanowsky is playing Keith Smithson in a three-game chess match. Winning a game scores 1 match point. Drawing a game scores 0.5 match point. After three games are played, the player with more match points is declared the champion. If the two players are tied after two games, they continue until someone wins a game.

During each game, Ulanowsky can play one of two ways: boldly – (45% chance of winning the game and a 55% chance of losing the game) or conservatively (90% chance of drawing the game and a 10% chance of losing the game.)

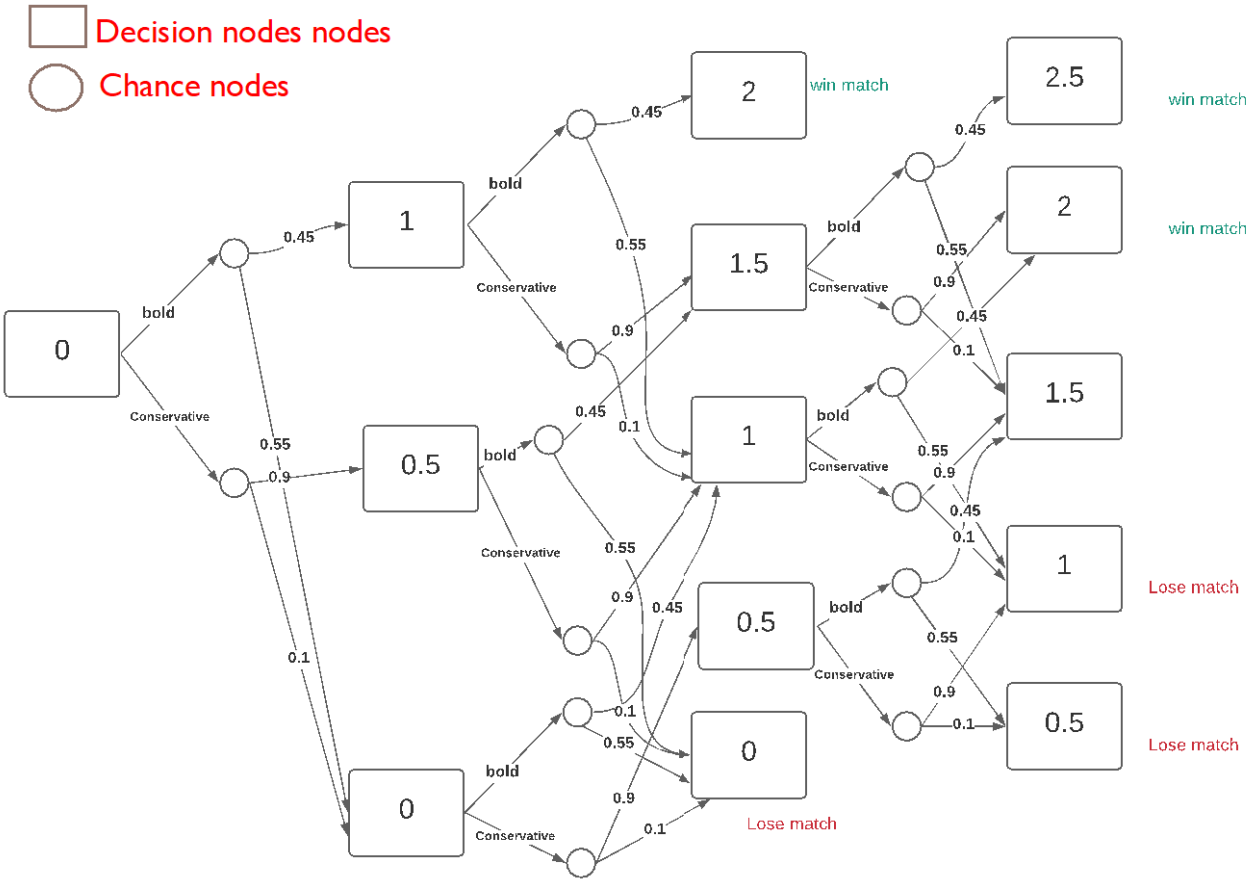
Ulanowsky's goal is to maximize his probability of winning the match. Use dynamic programming to help him accomplish this goal.

Answer

The DP recursion is defined as follows:

- Stages (t): Games (t=1,2,3,4)
- State (i): points of Ulanowsky at the beginning of game t.
- Decision: Playing boldly or conservatively
- $f_t(i)$: Maximum probability of winning the match if Ulanowsky has I points at the beginning of game t.
- $f_t(i) = Max \begin{Bmatrix} Bold & 0.45 \cdot f_{t+1}(i+1) + 0.55 \cdot f_{t+1}(i) \\ Conservative & 0.9 \cdot f_{t+1}(i+0.5) + 0.1 \cdot f_{t+1}(i) \end{Bmatrix}$
- Given:
  - $f_4(0) = f_4(0.5) = f_4(1) = 0$
  - $f_4(1.5) = 0.45$
  - $f_4(2) = f_4(2.5) = f_4(3) = 1$
- Find $f_1(0)$

A decision tree can be built to illustrate the model:



The probabilities of winning the match can be calculated starting from the last stage.

7.4 Further Examples of Probabilistic Dynamic Programming Formulations

Many probabilistic dynamic programming problems can be solved using recursions of the following form (for max problems):

$$f_t(i) = \max_a \left\{ (\text{expected reward during stage } t | i, a) + \sum_j p(j|i,a,t) f_{t+1}(j) \right\}$$

Where $f_t(i)$ is the maximum expected reward that can be earned during stages t, t+1,… end of the problem, given that the state at the beginning of stage t is i. The max in the formula is taken over all actions a that are feasible when the state at the beginning of stage t is i. $p(j|i, a, t)$ is the probability that the next period's state will be j, given that the current (stage t) state is i and action a is chosen. Hence, the summation in the formula represents the expected reward from stage t+1 to the end of the problem. By choosing a to maximize the right-hand side of equation we are choosing a to maximize the expected reward earned from stage t to the end of the problem, and this is what we want to do.

Example 7.6. Sunco Oil Drilling *(Winston 19.4, p. 1029)*

Sunco Oil has D dollars to allocate for drilling at sites 1,2,…,T. If x dollars are allocated to site t, the probability is $q_t(x)$ that oil will be found on site t. Sunco estimates that if site t has any oil, it is worth r_t dollars. Formulate a recursion that could be used to enable Sunco to maximize the expected value of all oil found on sites 1,2,…,T.

Answer

The stage should represent the number of sites, the decision for site t is how many dollars to allocate to site t, and the state is the number of dollars available to allocate to sites t, t+1,…,T. We therefore define $f_t(d)$ to be the maximum expected value of the oil that can be found on sites t, t+1,…,T if d dollars are available to allocate to sites t, t+1,…,T. We make the reasonable assumption that $q_T(x)$ is a nondecreasing function of x. If this is the case, then at stage T, all the money should be allocated to site T. This yields;

For $t = T$ $\quad f_T(d) = r_T q_T(d) + \big(1 - q_T(d)\big)0 = r_T q_T(d)$

For $t < T$, $\quad f_t(d) = \max_x \{r_t q_t(x) + f_{t+1}(d - x)\}$

where x must satisfy $0 \le x \le d$. The last recursion follows, because $r_t q_t(x)$ is the expected value of the reward for stage t, and since Sunco will have $d - x$ dollars available for sites t+1, t+2,…,T, $f_{t+1}(d - x)$ is the expected value of the oil that can be found by optimally drilling at sites t+1, t+2,…,T. To solve the problem, we would work backward until $f_1(D)$ had been determined.

Example 7.7. Waiting in Line *(Winston 19.4, p. 1030)*

When Sally Mutton arrives at the bank, 30 minutes remain on her lunch break. If Sally makes it to the head of the line and enters service before the end of her lunch break, she earns reward *r*. However, Sally does not enjoy waiting in lines, so to reflect her dislike for waiting in line, she incurs a cost *c* for each minute she waits. During a minute in which *n* people are ahead of Sally, there is a probability *p*(*x*|*n*) that *x* people will complete their transactions. Suppose that when Sally arrives, 20 people are ahead of her in line. Use dynamic programming to determine a strategy for Sally that will maximize her expected net revenue (reward – waiting costs).

Answer

When Sally arrives at the bank, she must decide whether to join the line or to give up and leave. At any later time, she may also decide to leave if it is unlikely that she will be served by the end of her lunch break. If 1 minute remained, Sally's decision would be simple: She should stay in line if and only if her expected reward exceeds the cost of waiting for 1 minute (*c*). Then we can work backward to a problem with 2 minutes left, and so on. We define $f_t(n)$ to be the maximum expected net reward that Sally can receive from time *t* to the end of her lunch break if at time *t*, *n* people are ahead of her. We let *t* = 0 be the present and *t* = 30 be the end of the problem. Since *t* = 29 is the beginning of the last minute of the problem, we write

$$f_{29}(n) = max\begin{Bmatrix} 0 & (Leave) \\ rp(n|n) - c & (Stay) \end{Bmatrix}$$

This follows because if Sally chooses to leave at time 29, she earns no reward and incurs no more costs. On the other hand, if she stays at time 29, she will incur a waiting cost of *c* (a revenue of – *c*) and with probability *p*(*n*|*n*) will enter service and receive a reward *r*. Thus, if Sally stays, her expected net reward is *rp*(*n*|*n*) – *c*.

For *t* < 29, we write

$$f_t(n) = max\begin{Bmatrix} 0 & (Leave) \\ rp(n|n) - c + \sum_{k<n} p(k|n) f_{t+1}(n-k) & (Stay) \end{Bmatrix}$$

The last recursion follows, because if Sally stays, she will earn an expected reward (as in the *t* = 29 case) of *rp*(*n*|*n*) – *c* during the current minute, and with probability *p*(*k*|*n*), there will be *n* – *k* people ahead of her; in this case, her expected net reward from time *t* + 1 to time 30 will be $f_{t+1}(n - k)$. If Sally stays, her overall expected reward received from time *t*+1, *t*+2,…, 30 will be

$$\sum_{k<n} p(k|n) f_{t+1}(n-k)$$

Of course, if *n* people complete their transactions during the current minute, the problem ends, and Sally's future net revenue will be zero.

To determine Sally's optimal waiting policy, we work backward until $f_0(20)$ is computed. If $f_0(20)$ is attained by "stay," Sally stays and sees how many people are ahead of her at time 1. She continues to stay until a situation arises for which the optimal action is "leave" or she begins to be served. In either case, the problem terminates.

Problems in which the decision maker can terminate the problem by choosing a particular action are known as **stopping rule problems;** they often have a special structure that simplifies the determination of optimal policies.

Example 7.8. Cash Management Policy *(Winston 19.4, p .1031)*

E. J. Korvair Department Store is trying to determine an optimal cash management policy. During each day, the demand for cash may be described by a random variable **D**, where *p*(**D** = *d*) = *p*(*d*). At the beginning of each day, the store sends an employee to the bank to deposit or withdraw funds. Each bank transaction costs *K* dollars. Then E. J.'s demand for cash is met by cash left from the previous day plus money withdrawn (or minus money deposited). At the end of the day, the store determines its cash balance at the store. If the cash balance is negative, a shortage cost of *s* dollars per dollar short is incurred. If the ending balance is positive, a cost of *i* dollars per dollar held is incurred (be-cause of loss of interest that could have been earned by depositing cash in the bank). At the beginning of day 1, the store has $10,000 cash on hand and a bank balance of $100,000. Formulate a dynamic programming model that can be used to minimize the expected cost of filling the store's cash needs for the next 30 days.

Answer

To determine how much money should be withdrawn or deposited, E. J. needs to know its cash on hand and bank balance at the beginning of the day. As usual, we let time be the stage. At the beginning of each stage (or day), E. J. must decide how much to withdraw from or deposit in the bank. We let *ft* (*c*, *b*) be the minimum expected cost incurred by the store during days *t*, *t*+1,…,30, given that at the beginning of day *t*, the store has *c* dollars cash at the store and *b* dollars in the bank.

We observe that

$$f_{30}(c,b) = \min_x \left\{ K\delta(x) + \sum_{d \le c+x} p(d)(c+x-d)i + \sum_{d \ge c+x} p(d)(d-c-x)s \right\}$$

Here, x is the amount of money transferred from the bank to the store (if $x < 0$ money is transferred from the store to the bank). Since the store cannot withdraw more than b dollars from the bank or deposit more than c dollars in the bank, x must satisfy $b \geq x \geq -c$. Also, in the equation, $\delta(0) = 0$ and $\delta(x) = 1$ for $x \neq 0$. In short, $K\delta(x)$ picks up the transaction cost (if there is a transaction). If $d \leq c + x$, the store will end the day with $c + x - d$ dollars, so a cost of $i(c + x - d)$ is incurred (because of lost interest). Since this occurs with probability $p(d)$, the first sum in the equation represents the expected interest costs incurred during day 30. Also note that if $d \geq c + x$, the store will be $d - c - x$ dollars short, and a shortage cost of $s(d - c - x)$ will be incurred. Again, this cost is incurred with probability $p(d)$. Hence, the second sum in the equation is the expected shortage cost incurred during day 30.

For $t < 30$, we write

$$f_t(c,b) = \min_x \left\{ K\delta(x) + \sum_{d \leq c+x} p(d)(c + x - d)i + \sum_{d \geq c+x} p(d)(d - c - x)s + \sum_d p(d) f_{t+1}(c + x - d, b - x) \right\}$$

As in the previous equation, x must satisfy $b \geq x \geq -c$. Also, the term $K\delta(x)$ and the first two summations yield the expected cost incurred during day t. If day t demand is d, then at the beginning of day $t + 1$, the store will have $c + x - d$ dollars cash on hand and a bank balance of $b - x$. Thus, with probability $p(d)$, the store's expected cost during days t+1, t+2,…,30 will be $f_{t+1}(c + x - d, b - x)$. Weighting $f_{t+1}(c + x - d, b - x)$ by the probability that day t demand will be d, we see that the last sum in the formula is the expected cost incurred during days t+1, t+2,…,30. Hence, the formula is correct. To determine the optimal cash management policy, we would use both equations to work backward until $f_1(10{,}000, 100{,}000)$ has been computed.

References

Bazaraa, M. S., Jarvis, J. J., & Sherali, H. D. (2010). Linear Programming and Network Flows. 4th Edition. John Wiley & Sons

Rardin R.L. (1998) "Optimization in Operations Research", Prentice Hall Inc.

Render B., Stair R.M. Jr., Hanna M.E. (2003) "Quantitative Analysis for Management", Pearson Education Inc.

Taha H.A. (2000) "Yoneylem Arastirmasi", Literatur Yayincilik (cev. Alp Baray and Sakir Esnaf)

Taha H.A. (2003) "Operations Research: An Introduction", Pearson Education Inc.

Taylor B.W. III (2002) "Introduction to Management Science", Pearson Education Inc

Walker R.C. (1999) "Introduction to Mathematical Programming", Prentice Hall Inc.

Williams, H. P. (2013). Model building in mathematical programming. John Wiley & Sons.

Winston W.L. (2004) “Operations Research: Applications and Algorithms”, Brooks/Cole – Thomson Learning

Winston W.L., Albright S.C. (2001) "Practical Management Science", Duxbury Press, Wadsworth Inc.