MUHAMMAD AYAZ HASAN

20K-1044

SE-4A

ASSIGNMENT #3

Operation Research

Q.1

~~B~~ K = Keep it.
R = Replace it.

Stage 4:

	K	R	Optimal Solution	
t	r(t)+s(t+1)-c(t)	r(0)+s(t)+s(1)-c(0)-I	f(t)	Dicision
1	19+60-0.6=78.4	20+80+80-0.2-100=79.8	79.8	R
2	18.5+50-1.2=67.3	20+60+80-0.2-100=59.8	67.3	K
3	17.2+30-1.5=45.7	20+50+80-.2-100=49.8	49.8	R
4	Must be replaced	20+5+80-.2-100=48	48	R

Stage: 3

	K	R	Optimal Sol	
t	r(t)-c(t)+f(t+1)	r(0)+s(t)-c(0)-I+f(1)	f(t)	Decision
1	19-.6+67.3=85.7	20+80+0.2-100+79.8=79.6	85.7	K
2	18.5-1.2+49.8=67.1	20+60-0.2-100+79.8= 59.6	67.1	K
3	14-1.8+48=17	20+10-0.2-100+79.8=9.6	19.6	R

Stage 2

	K	R	Optimal Sol	
t	r(t)-c(t)+f(t+1)	r(0)+s(t)-c(0)-I+f(1)	f(t)	Decision
1	19-0.6+67.1=85.5	20+80-0.2-100+85.7 =85.5	85.5	K OR R
4	15.5-1.7+19.6=33.4	20+30-0.2-100+85.5=35.5	35.5	R

Stage 1

	K	R	Optimal Sol	
t	r(t)-c(t)+f(t+1)	r(0)+s(t)-c(0)-I+f(1)	f(t)	Decision
3	17.2-1.5+35.5=51.2	20+50-0.2-100+85.5= 55.3	55.3	R

Scanned with CamScanner

Q. 2

i) Dynamic Programming:

Dynamic Programming is a process to solve optimization problem.

In Software development projects, dynamic programming uses an algorithm that break down complex coding problems into subproblem.

Characteristics:-

1). Subproblems Overlap:

2) Substructure has optimal Property

Example:

Top-down example: Apply the Fibonacci sequence, where each number in the series represent the sum of first two preceding numbers.

Deterministic Dynamic Programming:

determine the optimum solution to an n-variable problem by decomposing it into n stages with each stage constituting a single-variable sub problem.

ii) Integer Programming:

If requiring integer values is the only way in which a Problem deviates from a linear programming formulation, then it is an integer Programming.

Prototype Example:

The CALIFORNIA MANUFACTURING COMPANY is considering expension by building a new factory in either Los Angles or San Fransisco or perhaps even in both cities. It also is considering building at most one new ware house but the choice of the location is restricted to a city where a new factory is being built. The net present value of each of these alternatives is shown in the fourth colom of Table. The objective is to find the feasible combination of alternatives that maximize the total net present value.

$$x_j = \begin{cases} 1 & \text{if decision } j \text{ is yes} \quad (j = 1,2,3,4) \\ 0 & \text{if decision } j \text{ is no} \end{cases}$$

Decision No	Yes or NO Qs	Decision Variable	Net Present Value	capital Required
1	Build in L.A ?	x_1	\$9 M	\$6 M
2	Build in S.F?	x_2	\$5 M	\$3 M
3	Build in L.A?	x_3	\$6 M	\$5 M
4	Build in S F?	x_4	\$4 M	\$2 M

2m

2nst

(iii) Binary Integer Programming:

IP problems that contain only binary variables sometime are called binary Integer Programming

Application:

Investment Analysis:

L.P is used to make capital budgeting decisions about how much to invest in various project. However, as the California Manufacturing co. example demonstrate, some capital budgeting decision do not involve how much to invest but rather, whether to invest a fixed amount. Specifically, the four decision do not involve how much to invest, but rather whether to invest a fixed amount.

In general, capital budgeting decision, about fixed investment are yes-or-no decisions of the following type.

Each yes-or-no decision:

Should we make a certain fixed amount invest?

Its decision variable $= \begin{cases} 1 & \text{yes} \\ 0 & \text{No} \end{cases}$

iv) Perspective Approach on Solving Integer Programming.

It may seems that IP problems should be relatively easy to solve. After all linear programming problems can be solved extremely efficiently and the only difference is that IP problem have far fewer solutions to be considered. In fact, pure IP problems with bounded feasible region are guaranted to have just a fine number of feasible solutions. unfortunatly there are two facilities in this line of reasoning. One is having a finite no of feasible solution ensure that the problem is solvable.

For example, take a simple case of BIP problems with n variables, there are 2^n solution to be considered. Thus each ~~ca~~ time n is increased by 1, the no of solution is doubled. This pattern referred as the exponential growth of the difficulty of problem. With $n=10$ there are more than 1000 solutions; with $n=20$, there are more than 1000000; with $n=30$ there are more than 1 billion. The second fallacy is that removing some feasible sol from a Linear Progrg Problem will make it easier to solve.

v) Branch Cut Approach

It is common now for the branch-cut approach to solve some problem with many thousand variable and sometime even hundred of thousand of variable.

by incorporating & further developing the branch cut approach, striking improvements in linear programming algorithm that are heaviy used within the BIP algorithm & the great speed-up in computers.

This approach cannot consistently solve all pure BIP Problem with few thousand variables.

The vary large pure BIP problems solved have sparse A matrix e.g: the percentage of coefficient in the functional constraints that are nonzeroes is quite small. In fact this approach dependent heavily on ~~up~~ this sparsity.

vi) The constraint Programming:

In constraint programming, a problem is viewed as a series of limitation on what could possibly be a valid solution. This paradigm can be applied to effectively solve a group of problem that can be translated to variable and constraints or represented as mathematic equation.

Example:

let's take pythogoras theorem $a^2+b^2=c^2$. The constraint is represented by its equation which has three variables & each has a domain.

$$c = \sqrt{a^2+b^2}$$
$$a = \sqrt{c^2-b^2}$$
$$b = \sqrt{c^2-a^2}$$

These function satisfy the main constrant & check the domains each of the above function should validate the input.