MAT168 Project 2

Xing Yang Lan; SID: 915113655

1. Computational Equivalence of Linear Optimization and Linear Feasibility. (10 points)

1. Minimizing a linear function with linear constraints exists in polynomial time as the incremental and finite constraints can be considered hyperplanes cutting into the feasible region of a polytopes feasible region. Linear inequalities are therefor bounded within polynomials, and can always be solved in $O(n^3)$, by simplex.
2. If there existences a solution to the minimization of a system of linear inequalities, then that solution stands for such a system of linear inequalities, so a) can decide b).
3. Otherwise, if the system of linear inequalities with linear constraints is not feasible, the same system correspondingly is not feasible and has no solution. The 1:1 mapping makes a) and b) equivalent.

2. Another Way to do Dynamic Programming for Knapsack Problems (10 points)

The problem is transformable to a recursive minimization on weight, with the incremental recursion occurring in this case; when the cost of the current k-items is above the C lower-bound.

$$H[k,w,c] \begin{cases} H[k-1, w_{k-1}, c_{k-1}] & \text{if any } c_k > c \\ \min\limits_{w} \{ H[k-1, w_{k-1}, c_{k-1}], [H[k-1, w-w_k, c-c_k]] \end{cases}$$

Asymptotically for items n, running the function will call on itself at most $(2^n + 2^n - 1)$ times The recursion therefore has time complexity $O(2^n)$ and space complexity $O(n)$.

7. The inequalities (10 points)

After modifying x1 and x2 into x1+, x1-, x2+, x2-, and an additional slack variable, the reduced row echelon forms are for both system 1 and 2, denoted S1, S2 are:

1.0000	-1.0000	0	0	0
0	0	1.0000	-1.0000	0
0	0	0	0	1.0000

Then, rref([S1,eye(3,3)]) gives the desired invertible 3x3 matrix for transformation from S1 to S2 as:

0.3333	-2.3333	2.0000
0.6667	-2.6667	1.0000
0.0067	-0.0267	0.0200

----

4. An Approximation for Max Cut. (10 points)

-

This could be done recursively as.

$b_k = 0$

$$C(k,b) \begin{cases} \int_{b_{k-1}}^{b} c_{k-1}, (b - b_{k-1}) < b_{k-1} \\ \int_0^1 c_k\{\max(x_{k-1}, \frac{v_{k-1}}{c_{k-1}})\} + b_k \end{cases}$$

$x \in R^n$

such that for $x \in R^n$ for $X_k = x_i$ & $X_{k-1} = y_i$, X_k is optimal when

$\varepsilon \in R^n$ $\quad \varepsilon \geq \frac{v_i}{c_i} - \frac{y_i}{c_i} \geq 0$

$y_i > 0 \Rightarrow X_{k-1} = y_i \Rightarrow X_k = x_i = i$

$x_i > 0 \Rightarrow \frac{y_i}{c_i} = \frac{v_i}{c_i} + \varepsilon.$

b) when $\frac{v_i}{c_i} > \varepsilon$ then k, $\max(X_{k-1}, \frac{v_{k-1}}{c_{k-1}})$ returns i. if

if $c_i = 0$, ROI or $\frac{v_i}{c_i} = \infty$

c) When $\frac{v_i}{c_i} < \varepsilon$ then k for $\max(x_{k-1}, \frac{v_{k-1}}{c_{k-1}}) \neq i$

d) if, $\sum_{i \in I} c_i + c_j \leq b$ then, $\left\{ \max \sum_{i \in I}^{j} v_i x_i = v_i x_i + v_j x_j ; x_i, x_j = 1 \right.$

for all c_i, w/ b_k = sum costs

$b - b_k > 1$, for j; $b - b_k > 1$, x_j is integrated $[0,1]$

e) when $c_i x_i + c_{i+1} x_{i+1} + \dots c_{j-1} x_{j-1} + c_j x_j \neq b$; and $\sum_{i \in I} c_i + c_j > b$,

then

$b - \sum_{i \in I} c_i > 1 = x_i$; x_i for $i \in I = 1$; and $\frac{b - \sum_{i \in I} c_i}{c_j} = x_j$ for the remaining fraction of a purchase of x.

f. Select x_k with highest ROI and integrate $[0,1]$ along its cost, until $b - b_{k-1} \leq b_{k-1}$

then, integrate $\int_{b_{k-1}}^{b} c_{k-1}$

5. Fractional Knapsack Problem (10 points)

As both cost and value can be increased incrementally for every item, the optimal solution will contain as many of the items with the items with the highest rates of return (ROI) as the budget constraint will allow.

This could be done recursively as.

$b_k = 0$

$$C(k,b) \begin{cases} \int_{b_{k-1}}^{b} c_{k-1}, (b - b_{k-1}) < b_{k-1} \\ \int_0^1 c_k\{\max(x_{k-1}, \frac{v_{k-1}}{c_{k-1}})\} + b_k \end{cases}$$

$x \in R^n$

such that for $x \in R^n$ for $X_k = x_i$ & $X_{k-1} = y_i$ X_k is optimal when

$\varepsilon \in R^n$ $\quad \varepsilon \geq \frac{v_i}{c_i} - \frac{y_i}{c_i} \geq 0$

$y_i > 0 \Rightarrow X_{k-1} = y_i \Rightarrow X_k = x_i = i$

$x_i > 0 \Rightarrow \frac{y_i}{c_i} = \frac{v_i}{c_i} + \varepsilon.$

b) when $\frac{v_i}{c_i} > \varepsilon$ then k, $\max(X_{k-1}, \frac{v_{k-1}}{c_{k-1}})$ returns i.

if $c_i = 0$, ROI or $\frac{v_i}{c_i} = \infty$

c) When $\frac{x_i}{c_i} < \varepsilon$ then k for $\max(x_{k-1}, \frac{v_{k-1}}{c_{k-1}}) \neq i$

d) if, $\sum_{i \in I} c_i + c_j \leq b$ then, $\max \sum_{i \in I}^{j} v_i x_i = v_i x_i + v_j x_j$; $x_i, x_j = 1$

for all c_i, w/ b_k = sum costs

$b - b_k > 1$, for j; $b - b_k > 1$, x_j is integrated $[0, 1]$

e) When $c_i x_i + c_{i+1} x_{i+1} + \dots c_{j-1} x_{j-1} + c_j x_j \ngtr b$; and $\sum_{i \in I} c_i + c_j > b$,

then $b - \sum_{i \in I} c_i > 1 = x_i$, x_i for $i \in I = 1$; and $\frac{b - \sum_{i \in I} c_i}{c_j} = x_j$ for the remaining fraction of a purchase of x.

f. Select x_k with highest ROI and integrate $[0, 1]$ along its cost, until $b - b_{k-1} \leq b_{k-1}$

then, integrate $\int_{b_{k-1}}^{b} c_{k-1}$