DAA Programming Assignment 2

Balaji Karedla

April 14, 2025

1 Problem Statement

There is am FMCG company, which produces multiple products. It needs to decide its sales budget for various products across various cities. For each pair of a product and a city, they are given a range (a lower bound and an upper bound) on the budget. They are also given a ranges for the total budget of any city and also for the total budget of any product. We need to figure out if there is a budget allocation which respects all the ranges. Abstractly, you have to fill in an $m \times n$ matrix X with integers, while satisfying the below constraints. Let $x_{i,j}$ be the (i, j) entry of the matrix.

- $l_{i,j} \leq x_{i,j} \leq u_{i,j}$ for every $1 \leq i \leq m, 1 \leq j \leq n$.
- $r_i \leq x_{i,1} + x_{i,2} + \cdots + x_{i,n} \leq R_i$ for every $1 \leq i \leq m$.
- $c_j \leq x_{1,j} + x_{2,j} + \cdots + x_{m,j} \leq C_j$ for every $1 \leq j \leq n$.

Here $\{l_{i,j}, u_{i,j}\}, \{r_i, R_i\}, \{c_j, C_j\}$ are given as input. You have to output

- whether it is possible to fill in the matrix (output 1 if possible otherwise 0).
- if possible what is the maximum possible total budget $\sum_{i,j} x_{i,j}$.
- if possible what is the minimum possible total budget $\sum_{i,j} x_{i,j}$.

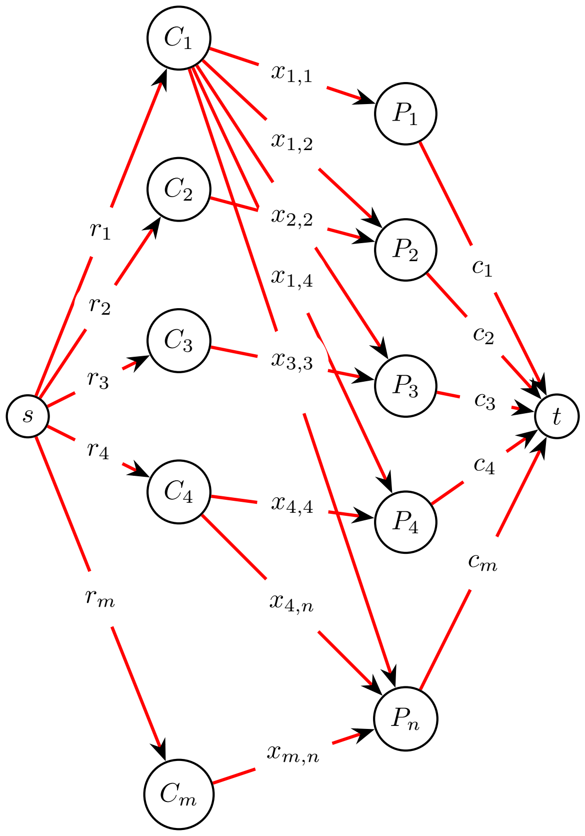
2 My Idea

We can change the problem a bit by adding a flow of $l_{i,j}$ to all the entries in the matrix, so now the problem changes to

- $0 \leq x_{i,j} \leq (u_{i,j} - l_{i,j})$ for every $1 \leq i \leq m, 1 \leq j \leq n$.
- $\max\{0, (r_i - \sum_{j=1}^{n} l_{i,j})\} \leq x_{i,1} + x_{i,2} + \cdots + x_{i,n} \leq (R_i - \sum_{j=1}^{n} l_{i,j})$ for every $1 \leq i \leq m$.
- $\max\{0, (c_j - \sum_{i=1}^{m} l_{i,j})\} \leq x_{1,j} + x_{2,j} + \cdots + x_{m,j} \leq (C_j - \sum_{i=1}^{m} l_{i,j})$ for every $1 \leq j \leq n$.

Let the new lower bounds and upper bounds be defined with the same notation but with a dash.

If any of the upper bounds are < 0, there exists no solution to the problem and we can return 0.



Network flow representation of the problem (All edges $C_i \to P_j$ are not shown)

Now to represent the problem in network flow, we can take a source node and a sink node and m nodes that represent the cities and n nodes that represent the products. There exists nodes

- from s to C_i whose flow represents $\sum_{j=1}^{n} x_{i,j}$ which has the new lower bound r'_i and upper bound R'_i for every $1 \leq i \leq n$.
- from C_i to P_j whose flow represents $x_{i,j}$ which has the new lower bound 0 and upper bound $u'_{i,j}$ for every $1 \leq i \leq m$ and $1 \leq j \leq n$.
- from P_j to t whose flow represents $\sum_{i=1}^{m} x_{i,j}$ which has the new lower bound c'_j and upper bound C'_j for every $1 \leq j \leq n$.

My idea was to find (or show the infeasibility of) a flow that satisfies the constraintsand then run max-flow and min-flow to get the maximum and minimum flows. The problem can be represented as a network flow problem.

This has the minimum bound only on two sets of edges. Now if there exists a flow which satisfies the problem, there exists a solution to the problem by matching the $x_{i,j}$'s.

If there exists a solution to the problem, the solution to the problem, the solution to the maximum flow of the problem with constraints

- All the edges $s \to C_i$ have capacity r'_i.
- All the edges $C_i \to P_j$ have capacity $u'_{i,j}$.
- All the edges $P_j \to t$ have capacity C'_j

should be $\sum_{i=1,m} r'_i$ because there exists a solution with such a minimum.

If the maximum flow is less than $\sum_{i=1,m} r'_i$, we can return 0.

Now for all the edges $s \to C_i$, change the residual capacities by making saving this flow and making the minimum 0 and maximum to $R'_i - r'_i$

If every capacity of edges $P_j \to t$ is in the range expected, skip this step.

For all the flows in $P_j \to t$, which are less than c'_j, set the maximum capacity to c'_j and run the maximum flow with the current flow and the other edges' minimum flow to c'_i (by changing the residual flow of the backward edges). If the flow doesn't reach c'_i after running the maximum flow, return 0.

Now, change the minimum and maximum to c_j and C_j appropriately, and run the maximum flow. We get the maximum flow with the known constraints.

Now change all the constraints to negative, we can maximize the negative constraints to get the minimum flow.

3 Complexity Analysis

The complexity of the solution is $O(VE^2)$ as it is just Edmond's Karp algorithm[1] being run four times. The complexiy in terms of m and n is $O((m+n+2)(mn+m+n)^2) = O(m^2n^2(m+n))$

4 Compilation Instructions

To compile, run `g++ --std=c++20 23b1029.cpp`

References

[1] E maxx team. Edmonds-karp algorithm. https://cp-algorithms.com/graph/edmonds_karp.html, n.d.