Computational Methods And Optimization Project Abstract

Group 11: ANAGHA VASISTA, CHAITANYA MODI, MANAN CHAWLA, PRATHAM ARORA

Project: Capacitated Vehicle Routing Problem (CVRP) With Time Constraints

Updates:

Work done:

- We went through some websites, research papers and YouTube videos (kindly find them in references) to understand how to work through our project problem.
- Two of our team members (Anagha Vasista and Chaitanya Modi) learnt to work on Git Hub.

Work under progress:

- We are going through VRP to understand the basics through the VRP code using OR tools and Python.
- We are currently working on understanding the Mixed Integer Linear Program and also the code to implement our project idea.

Further work:

- We hope to complete going through the entire learning phase for the project by next week and start implementing our solution.

Other updates:

- We are going to look into Google Maps API for using real-world locations, distances and routes.

References:

1. https://satpal-singh.medium.com/solving-cvrptw-using-or-tools-in-python-214247f6d680
2. https://developers.google.com/optimization/routing/vrp
3. https://youtu.be/v9tUEsHD6BE?feature=shared
4. https://www.youtube.com/playlist?list=PLaoe2MTbJBvpFPyMMSOB-WrHofdHo3e74

Computational Methods And Optimization Project Abstract

Group 11: Anagha Vasista, Chaitanya Modi, Manan Chawla, Pratham Arora

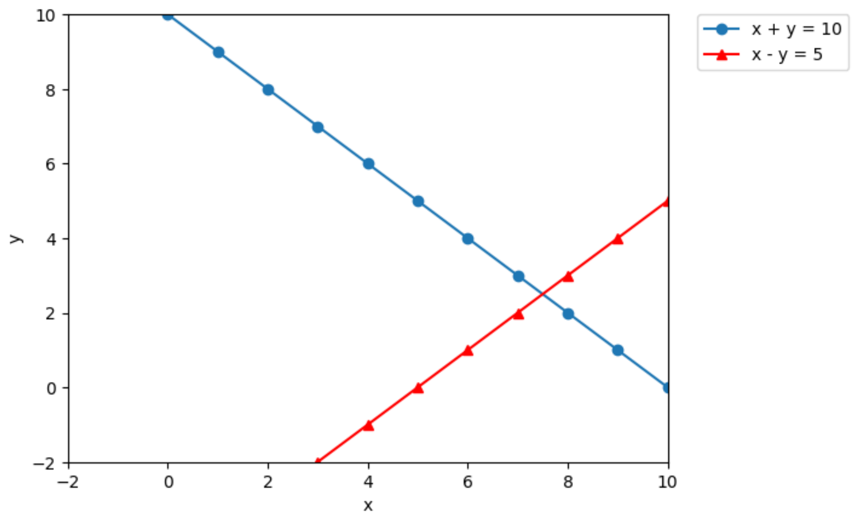
Capacitated Vehicle Routing Problem (CVRP) With Time Constraints

10th November, 2023

Updates:

This week we learnt the MILP problem and solved a few examples. In the update, we have illustrated the Knapsack Problem (using Tabular/Simplex method) as an example of a MILP. We also learnt Branch & Bound method, and dynamic programming as methods to solve MILPs .

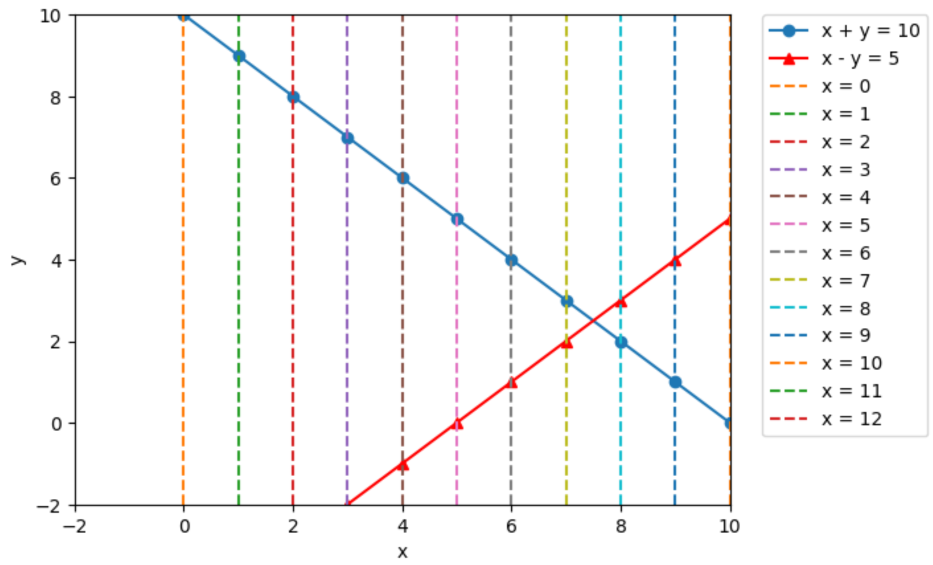
What is Linear Programming?



$x_i \in \Re^n$ ie x can take any rational value.

So, this above equation has a Solution at (7.5, 2.5)

What is Mixed Integer Linear Programming?



$x_i \in \{Z, \mathfrak{R}\}^n$ ie x can take any integer value.

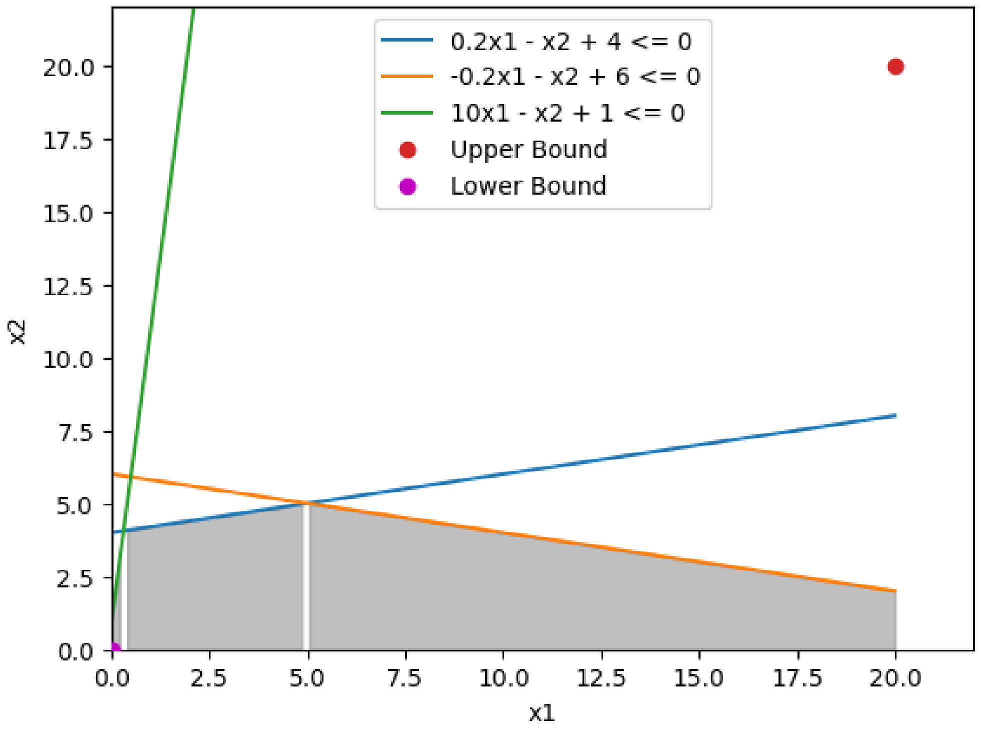
Now that there exists another constraint that the feasible solutions can only be integer values the same set of equations no longer have a solution because now the intersection or the feasible solutions should lie on the dashed lines.

Difference between LP and MILP:

An LP (linear program) involves minimising (or maximising) a linear function subject to linear constraints on the variables. Any solution that satisfies the constraints is feasible. A MILP is an LP with the addition of integrality restrictions on some or all of the variables.
At first glance MILPs look easier to solve because the finite set of feasible solutions is much smaller than the uncountably infinite number of solutions for an LP.
But, when you take a closer look at the feasible region to be optimised, you can see that for an LP, you only need to look at the vertices of the feasible region. The simplex method does just that, and therefore can solve LPs more efficiently than any known algorithm for MILPs.
The barrier method also solves LPs, and moves through the interior of the feasible region rather than along the boundary. But it too has a way to efficiently move from one feasible solution to another. This is much more challenging for MILPs, and the branch and bound algorithms currently in favour use binary search trees that can grow exponentially.

Some basic/common ways to solve MILP:

1. **Simplex Method**



Way to Solve MILPs by this method:
1. Initial step:
Start in a feasible basic solution at a vertex.
2. Iterative step:
Move to a better feasible basic solution at an adjacent vertex
3. Optimality test:
A feasible basic solution at a vertex is optimal when it is equal or better than feasible basic solutions at all adjacent vertices.

2. Branch & Bound Method

$x_1 + x_2 \leq 50$

$4x_1 + 7x_2 \leq 280$

$x_1, x_2 \geq 0$

The first step is to relax the integer constraint. We have two extreme points for the first equation that form a line: [x1 x2] = [50 0] and [0 50]

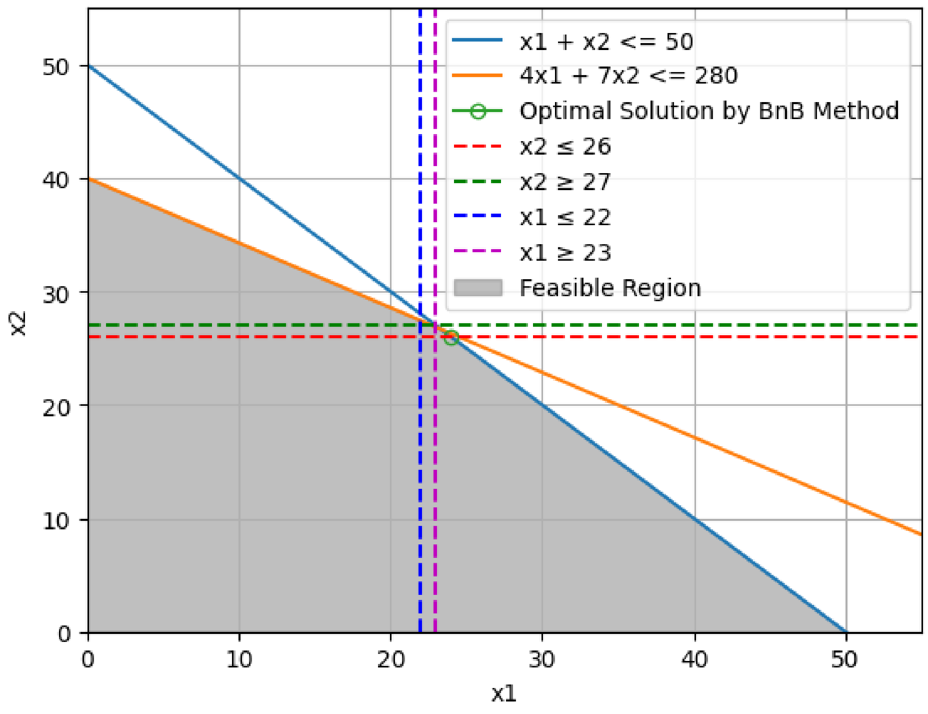
We get the second line with the vector points [0 40] and [70 0].

The third point is [0 0].

This is a convex hull region so the solution lies on one of the vertices of the region. We can find the intersection using row reduction, which is [70/3 80/3] with a value of 276.667

We test the other extreme points by sweeping the line over the convex region and find this is the maximum over the Reals.

We choose the variable with the maximum fractional part, in this case x2 becomes the parameter for the branch and bound method. We branch to x2 ≤ 26 and obtain 276 at (24, 26). We have reached an integer solution so we move to the other branch x2 ≥ 27 and obtain 275.75 at (22.75, 27). We don't have an integer solution so we branch x1 to x1 ≤ 22 and we find 274.571 (22, 27.4286). We try the other branch x1 ≥ 23 and there are no feasible solutions. Thus, the maximum is 276 with x1 at 24 and x2 at 26.



<u>Knapsack Problem (0-1 Method):</u>
This is the tabular method for solving the Knapsack Problem.

Example :
Suppose we have the following items with their values (v_i) *and weights* (ω_i) :

Item	Value	Weight
1	3	2
2	4	3
3	5	4
4	6	5

And the maximum weight capacity (W) *of the knapsack is* 5 *and total items* (n) *is* 4

Tabular Method :

Create a table with rows representing items (*from* 0 *to* 4) *and columns representing the knapsack's weight capacity* (*from* 0 *to* 5). *Initialize the table with zeros. terate through each item and each possible knapsack weight capacity, updating the table based on the optimal value.*

The entry $T[i][j]$ *represents the maximum value that can be obtained with the first i items and a knapsack capacity of j.*

$T[i][j] \;=\; max(T[i-1][j],\; v_i + T[i-1][j-\omega_i])$
If the weight of the current item (ω_i) *is greater than the current capacity* (j), *we simply copy the value from the cell above. Otherwise, we consider whether it's more valuable to include the current item or not.*

$$\begin{bmatrix} & 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 3 & 3 & 3 & 3 \\ 2 & 0 & 0 & 3 & 4 & 4 & 7 \\ 3 & 0 & 0 & 3 & 4 & 5 & 7 \\ 4 & 0 & 0 & 3 & 4 & 5 & 7 \end{bmatrix}$$

To identify the items that must be put into the knapsack to obtain the maximum profit, start from the bottom – right corner of the table and backtrack to reconstruct the selected items.
If $T[i][j]$ *is equal to* $T[i-1][j]$, *the item i was not selected. Otherwise, it was selected. In this case,* 7 *is in item* 2 *and not in item* 1, *so item* 2 *will be selected. Then we look for* $T[i][j] - i[v_i]$.
$7 - 4 = 3$ *and repeat the process.*
After all entries are scanned, the marked labels represent the items that must be put into the knapsack.
In this case the maximum possible value that can be put into the knapsack = 7. *We put items* 1 *and* 2 *intot he knapsack.*

References:

1. Guignard, M. M., & Spielberg, K. (1972). Mixed-integer Algorithms for the (0,1) Knapsack Problem. IBM Journal of Research and Development, 16(4), 424–430. doi:10.1147/rd.164.0424
2. http://www.or.deis.unibo.it/kp/Chapter2.pdf
3. https://en.wikipedia.org/wiki/Knapsack_problem
4. H4.pdf (tue.nl)
5. Using the Simplex Method in Mixed Integer Linear Programming (loria.fr)
6. https://www.math.wsu.edu/students/odykhovychnyi/M201-04/Ch06_1-2_Simplex_Method.pdf
7. 0/1 Knapsack Problem | Dynamic Programming | Example | Gate Vidyalay

Tasks for Week 3:

1. For next week, we will learn how VRP can be solved using the methods we learnt this week, and then move onto CVRP and work on the code to solve CVRP using Python and Google OR-tools.
2. We will also be solving and explaining the Knapsack problem using Python.