Decomposition Problems

COMP4691-8691
School of Computer Science, ANU

1 Complicating Variables and Constraints

- Explain what complicating variables and complicating constraints are. Relate them to Column Generation and Benders Decomposition.
  - **Complicating Constraints**: These constraints link variables in a way that makes the feasible region difficult to define or compute efficiently, often introducing complexity into the problem. For instance the maximum time to reach the goal in the constrained shortest path problem and deciding what which lengths to cut a given rod in the cutting stock problem.
  - **Complicating Variables**: These variables make solving the problem hard because they are not continuous and/or link several constraints together. Fixing the value of the complicating variables results in a much easier to solve optimisation problem. Examples of complicating variables are: (i) if facility k will be built ($y_k \in \{0, 1\}$) in the facility location problem; and (ii) if a bus stop i is a hub ($y_i \in \{0, 1\}$) and if there is a route between hubs i and j ($y_{ij} \in \{0, 1\}$) in the BusPlus example.

Column Generation is a technique to handle **complicating constraints**. The problem is usually reformulated so into an "simpler" problem with a very large number variables which is called the Master Problem. Column generation iteratively builds a new optimisation problem called Reduced Master Problem (RMP) by selecting which variables/columns to be added to it. To decide which variable to add to the RMP, a simpler optimisation problem is solved where the complicating constraint is not present. Instead, the dual variables associated with the complicating constraint in the RMP is used. This subproblem is known as the Pricing Problem.

Benders Decomposition is a technique to handle **complicating variables**. The is also decomposed into a Reduced Master Problem (RMP) and a subproblem. The RMP only contains the complicating variables of the original problem and a new variable η. The constraints in the RMP are the constraints involving only the complicating variables, optimality cuts and feasibility cuts. At each iteration, a subproblem is solved by fixing the values of the complicating variables to their optimal value in the current RMP and the result is at least one new optimality cut or feasibility cut to be added to the RMP.

- If we are solving an LP, what is the relationship between complicating variables and complicating constraints?

  For LPs, a complicating variable is the dual of a complicating constraint and vice-versa. That is, if in a primal LP there is a complicating constraint, we can apply Column Generation to it or apply Benders Decomposition to the dual LP. Another way of seeing this is looking at the constraint matrix A: a complicating variable is a column in A which is a row in $A^\top$, i.e., a constraint in the dual.

2 Column Generation

2.1 Vertex Coloring

In this question, we will revisit the vertex coloring problem you saw in COMP3620/6320 and solve it using Column Generation.

Given an undirected graph $G = (V, E)$, where V is the set of vertices and E the set of edges, a valid coloring of G assigns a color to all vertices V such that adjacent vertices have different color. More formally, for all edges $(i, j) \in E$, the color of $i \in V$ must be different from the color of $j \in V$. In this question, we want to find the **minimum number of colors** needed to generate a valid coloring of G.

- Formulate the vertex coloring problem as an ILP. Assume you are given a set C of available colors and use two types of variables: y_c to represent whether the color $c \in C$ is used or not; and x_{ic} to represent whether vertex $i \in V$ has color $c \in C$.

Let $\chi(G)$ be the **chromatic number** of G, i.e., the minimum number of colors required to color the graph, the ILP is formulated as:

$$
\begin{aligned}
\chi(G) = \min \sum_{c \in C} y_c & & (1)\\
\text{s.t.} \quad \sum_{c \in C} x_{ic} = 1 & \qquad \forall i \in V & (2)\\
x_{ic} + x_{jc} \leq 1 & \qquad \forall (i,j) \in E, \forall c \in C & (3)\\
x_{ic} \leq y_c & \qquad \forall i \in V, \forall c \in C & (4)\\
x_{ic} \in \{0,1\} & \qquad \forall i \in V, \forall c \in C & (5)\\
y_c \in \{0,1\} & \qquad \forall c \in C & (6)
\end{aligned}
$$

where:

(1). The objective function minimizes the total number of used colors.

(2). Each vertex $i \in V$ must be assigned exactly one color from C.

(3). Adjacent vertices $(i,j) \in E$ must not share the same color, for any $c \in C$.

(4). The variables x_{ic} and y_c are linked, ensuring that a vertex can only be assigned a color if that color is in use.

(5). x_{ic} is a binary variable that indicates whether vertex i is assigned color c.

(6). y_c is a binary variable that indicates whether color c is used in the coloring.

- Reformulate the problem as another ILP using a new set of variables that might be exponentially large (hint: look at the definition of *independent set*). The linear relaxation of this ILP will be our Master Problem.

Observations: Each color class forms an *independent set* in G.

We denote by $\mathcal{P}$ the set of (encodings of) all independent sets in G. Let $a_{ip} \in \{0,1\}$ be a parameter that denotes whether vertex $i \in V$ is contained in independent set $p \in \mathcal{P}$. The reformulated ILP is:

$$\chi(G) = \min \sum_{p \in \mathcal{P}} \lambda_p \tag{7}$$

$$\text{s.t.} \quad \sum_{p \in P} a_{ip}\lambda_p = 1 \qquad \forall i \in V \tag{8}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in \mathcal{P} \tag{9}$$

where:

(7). The objective function minimizes the number of sets used.

(8). Each vertex $i \in V$ must be covered by exactly one independent set $p \in \mathcal{P}$.

(9). λ_p is a binary variable that indicates whether independent set p is used.

Our Master Problem is the linear relaxation of the reformulated ILP:

$$\begin{aligned} &\min \sum_{p \in \mathcal{P}} \lambda_p \\ \text{s.t.} \quad &\sum_{p \in P} a_{ip}\lambda_p = 1 && \forall i \in V \\ &\lambda_p \in [0,1] && \forall p \in \mathcal{P} \end{aligned}$$

- What is the Reduced Master Problem?

  The Reduced Master Problem is the original problem with just a subset of variables. For our vertex coloring scenario, we use a subset of all independent sets $\mathcal{P}' \subseteq \mathcal{P}$ and obtain the following Reduced Master Problem:

$$\begin{aligned} &\min \sum_{p \in \mathcal{P}} \lambda_p \\ \text{s.t.} \quad &\sum_{p \in P} a_{ip}\lambda_p = 1 && \forall i \in V \\ &\lambda_p \in [0,1] && \forall p \in \mathcal{P}' \subseteq \mathcal{P} \end{aligned}$$

- What is the formulation of the pricing problem? Do you know the name of the problem being solved by the pricing problem?

  Let $\pi^t = (\pi_1, \dots, \pi_{|V|})$ denote the dual variables associated with the constraint $\sum_{p \in P} a_{ip}\lambda_p = 1$ for each vertex $i \in V$ in each iteration t. We use these dual variables to define the pricing problem. We aim to minimize the reduced cost, thus the pricing problem can be derived as:

$$
\begin{aligned}
\bar{c}^* = \min_{p \in \mathcal{P}} \bar{c}_p = \min_{p \in \mathcal{P}} 1 - (\pi_1, \dots, \pi_{|V|}) \cdot \begin{pmatrix} a_{1p} \\ a_{2p} \\ \vdots \\ a_{|V|p} \end{pmatrix} \\
= \min 1 - \sum_{i \in V} \pi_i x_i \\
= 1 - \max \sum_{i \in V} \pi_i x_i \\
\text{s.t.} \quad x_i + x_j \le 1 \quad \forall (i,j) \in E \\
x_i \in \{0,1\} \quad \forall i \in V
\end{aligned}
$$

  where x_i represents whether a vertex i is covered by a independent set we should construct.

  The problem being solved by the pricing problem is called a Maximum Weight Independent Set Problem. Note that this pricing problem is still hard to solve and we would need to use other techniques to make column generation efficient for this problem.

- Now you have the complete column generation approach to solve the Master Problem, i.e., the *linear relaxation* of the reformulated problem. Outline how you would continue by implementing a Branch-and-Price algorithm, i.e., combining Branch-and-Bound with column generation.

  The Branch-and-Price algorithm is the Branch-and-Bound algorithm when we use Column Generation to solve the linear relaxation in each node of the Branch-and-Bound tree. Thus in each node of the Branch-and-Bound tree, we will solve a similar linear relaxation as shown in the previous items for different variables λ_p being fixed. Here is the full outline of the algorithm:

  The Branch-and-Price Algorithm:

1. Solve the linear relaxation of the problem using Column Generation.
2. After solving the linear relaxation, check if the solution is integer. If the solution satisfies the integer constraints, it is feasible, and the current solution is an optimal integer solution.
3. If the solution is fractional, select a variable and that should be integer but is fractional, and create two branches:
    (a) One branch sets the variable to 0.
    (b) The other branch sets the variable to 1.
4. After branching, a new restricted master problem is solved at each node. Again, solve the pricing problem iteratively to generate additional columns, and update the master problem to get the new optimal solution.
5. Continue to branch at nodes until either a feasible integer solution is found or the node is pruned.
    - The branch is pruned if the solution to the linear relaxation at a node is infeasible, or the objective value of the solution is worse than the upper bound.
    - The upper bound is the best integer solution found so far, while the lower bound at any node is the value of the solution to the linear relaxation (solved by column generation) at that node.
6. The algorithm terminates when all nodes have been processed and the best feasible integer solution (upper bound) has been found. This solution is optimal for the original problem.

2.2 Cutting Stock Implementation

Implement the Cutting Stock example from the Column Generation lecture based on the file `cutting_stock.py`. You do not need to implement branch-and-bound, i.e., you only need to solve the linear relaxation of the original problem. For the pricing problem, solve the MIP problem shown in slide 27.

3 Benders Decomposition – Facility Location Implementation

In this question, you will finish modelling the Facility Location problem from the beginning of the Benders Decomposition lecture and implement a simplified version of it.

- Using the model in slide 5 of lecture 10, write the Benders Master Problem (BMP).

$$\begin{aligned}
\min \sum_k b_k y_k + \eta \\
\text{s.t.} \quad \eta \geq \sum_i (\pi_e[i] + \sum_k \pi_e[i,k] y_k) \qquad & \forall \pi_e \in E \\
0 \geq \sum_i (r_q[i] + \sum_k r_q[i,k] y_k) \qquad & \forall r_q \in Q \\
y_k \in \{0,1\} \quad \forall k
\end{aligned}$$

  Where $[i,k]$ and $[i]$ represents the index in the extreme point π_e and extreme ray r_q associated with original problem $x_{ik} \leq y_k$ constraint and $\sum_k x_{ik} = 1$ constraint. See two items below.

- The first simplification is that we will work with the (primal) sub-problem (SP) instead of the dual sub-problem (DSP). Formalize the (primal) sub-problem.

  Given a set of values $\hat{y}$ for the complicating variables y, the SP is:

$$\begin{aligned}
\min \sum_{ik} c_{ik} x_{ik} \\
\text{s.t.} \quad \sum_k x_{ik} = 1 \qquad & \forall i \\
x_{ik} \leq \hat{y}_k \qquad & \forall i,k \\
x_{ik} \in [0,1] \quad & \forall i,k
\end{aligned}$$

- We still need to the get the extreme points of the dual problem, i.e., the arg max of the DSP. How can you get it using the SP?

  We can get them by asking the solver to give us the associated optimal dual variables of the SP. Every modern solver will compute them almost for free. We can do the same for the extreme rays, that is, if the SP is infeasible we can ask the solver to compute an extreme ray in its dual for relatively cheap. Unfortunately, PuLP does not super the computing extreme rays but we will address this in the next item.

- The second simplification is that we will add a small modification to the BMP to guarantee that all the SPs are feasible; therefore we will not need to compute extreme rays to generate feasibility cuts. Write a simple constraint using only the y_k variables for the BMP that guarantees feasibility of the SPs. Explain your answer.

  The SP is infeasible only if no facility is built. As long as at least one facility is being built, then all the customers will have their demand attended since there is no maximum capacity for the facilities. Therefore, we can add to the BMP a constraint enforcing that we will build at least one facility, i.e., $\sum_k y_k \geq 1$.

- Now you have all the parts to implement the Benders decomposition approach for the Facility Location problem. Open the file `facility_location.py` and implement the missing methods. Here are some hints:

  – Place all the constraints in a python variable because you will need to access and manipulate them, for instance, `constr = Var('x') >= 10` then `model += constr`.

  – Since we will work on the SP as opposed to the DSP, you will need to change the constraints of the SP. Use the method `changeRHS`, e.g., `constr.changeRHS(-1)`.

  – To get the value of a dual variable associated to a constraint, use the attribute `pi`, e.g., `constr.pi`