ex5_holzberger

June 20, 2021

0.1 Exercise 11

To formulate the Bin Packing problem, such that we get exponentially many variables, we can formulate it as an instance of the set partitioning problem. The matrix $A \in \mathbb{B}^{n \times N}$ contains all N feasible bin packings in its columns. For example the j-th column could be $a_j = (0, 1, 0, ..., 0)^T$, stating that the bin corresponding to the j'th column contains only the second item. Moreover, since we only consider feasible packings, we have to ensure that the matrix A fullfils $\sum\limits_{i=1}^{n} l_i A_{i,j} \leq W \quad \forall j \in \{0, ..., n\}$. The ILP for the bin packing problem is then:

$$\min \sum_{j=1}^{n} x_j \tag{1}$$

$$\sum_{j=1}^{N} x_j A_{i,j} \geq 1 \quad (p_i)\ \forall i \in \{0, ..., n\} \tag{2}$$

$$x \in \mathbb{B}^N \tag{3}$$

Here the variables x_j are 1 if we choose a feasible packing in the column of A and 0 otherwise. The only constraint we have is that an element must be in at least one of the bins for all selected bins. For a column generation approach we start with the restricted master problem, that contains the matrix A', which has only a small subset of the columns of A but is other than that the same (we also start with only the variables x_j that are needed for the columns of A'). Important is that with the initial A' and its associated variables we can already get a feasible solution.

Next we generate columns in A' to improve on that solution by maximizing the reduced costs. For the reduced costs of a new variable x_j we want to generate a column $A_{i,j}$ that maximizes $\bar{c}_j = 1 - \sum\limits_{i=1}^{n} p_i A_{i,j}$ with the dual variable p_i associated to the constrained i. Note that any column that we generate needs to fulfill the capacity constraint from above, so the pricing problem can be formulated as a knapsack problem:

$$max \sum_{i=1}^{n} p_i a_i \tag{4}$$

$$\sum_{i=0}^{n} a_i l_i \leq W \tag{5}$$

$$a \in \mathbb{B}^n \tag{6}$$

Where we note that the generated column j is then $Ae_j = a_j$, given by the optimal solution of the knapsack problem. To see that the formulation above can produce a better LP-bound we consider the following instance: $L = \{2, 2, 2\}, W = 3, n = 3$. We consider the LP-relaxation of the standard formulation first. Here the optimal solution uses two bins since we can choose the variables as:

$$x = \begin{bmatrix} 1 & 1/2 & 0 \\ 0 & 1/2 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \tag{7}$$

Therefore the best LP-bound for the standard formulation is 2. Next consider our formulation with exponentially many variables. Here we first set up the matrix A of all feasible bins by:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{8}$$

Since we must fulfill the constraint $\sum_{j=1}^{N} x_j A_{i,j} \geq 1 \quad \forall i \in \{0, ..., n\}$, that is a linear combination of the columns of A and since the column are linear independent we need for the optimal solution $x = [1, 1, 1]^T$. This shows that for the optimal solution we need 3 bins and therefore the best bound of the LP-relaxation of this variant is 3, which is better than the bound that the standard-form gave us.

To solve the ILP by the Branch and Price approach one needs to be careful to not just branch on single x variables above, because this leads to a highly imbalanced branching tree. For example we set $x_0 = 1$ in the right branch and $x_0 = 0$ in the left branch. Then in the left branch we forbid that a bin is filled by the items corresponding to variable x_0. Therefore, the left branch still has exponential variables without fixing any bin and therefore is huge. For the right branch we have that $x_0 = 1$ meaning that one bin is already fixed and therefore we have significantly fewer choices to assign variables in this subtree. To generate balanced branches we want to branch on the variables of the original problem. Above we see that the bins are not enumerated, therefore, we cant branch by the choice if an item is assigned to a specific bin. We rather branch by the fact if two items must be placed in a bin together. Then in the left branch we set all $x_j = 0$ if the two items of choice are included in the assignment of the corresponding variable x_j and in the right branch $x_j = 0$ if they are not included in the assignment of the corresponding variable x_j. This will not increase the complexity knapsack problem that we solve for the pricing problem, since we can then solve that knapsack problem with the corresponding variables fixed to a value.

0.2 Exercise 12

First we replace every edge e by the arcs $(i, j), (j, i)$ to obtain a directed Graph $G = (V, A)$. A feasible pattern in this formulation corresponds to a feasible path from the root node $0 \in V$ to one of the terminal nodes $t \in T$. The set of feasible paths is given by $P := \{p \subset A : \text{p is simple path from 0 to t for } t \in T \text{ and } |p| \leq H\}$. We can encode all feasible paths p_k in the matrix $M \in \mathbb{B}^{|V|^2 \times |P|}$ by setting $M_{(i|V|+j),k} = 1$ if $(i, j) \in p_k$ and $M_{(i|V|+j),k} = 0$ otherwise. For ease of notation let $B_{i,j}^k := M_{(i|V|+j),k}$. Then we formulate the problem as:

$$\min \sum_{i,j} c_{i,j} x_{i,j} \tag{9}$$

$$\sum_k B_{i,j}^k \lambda_k \leq x_{i,j} m \quad (\alpha_{i,j})\ \forall (i,j \in A) \tag{10}$$

$$\sum_i \sum_k B_{i,t}^k \lambda_k \geq 1 \quad (\beta_t)\ \forall t \in T \tag{11}$$

$$\sum_i x_{i,j} \leq 1 \quad (\gamma_j)\ \forall j \in V \tag{12}$$

$$\sum_i x_{0,i} \leq 1 \quad (\nu) \tag{13}$$

$$\lambda_k \in \mathbb{B} \quad \forall k \in \{0, ..., |P|\},\ x_{i,j} \in \mathbb{B} \quad \forall i,j \in V \tag{14}$$

Here the paths are selected by the variable λ_k. The arcs that we select for our subgraph are selected by the variable $x_{i,j}$ that is coupled to the selected paths by the first constraint. Then the second constraint ensures that we have a path from the root to every terminal and the third and fifth constraints ensure that we obtain a tree. Note that the constant m is bigger or equal than the number of columns in M, which is equivalent to the number of paths from the root to a terminal. The formal definition of the pricing problem is then:

$$\bar{c}_k = \min_{k \in \{0,...,|P|\}} -\sum_i \sum_{t \in T} \beta_t B_{i,t}^k + \sum_{i,j} \alpha_{i,j} B_{i,j}^k \tag{15}$$

We note that for generating a feasible path $B_{i,t}^k$ always contains a terminal for some t. Therefore we can write:

$$\bar{c}_k = \min_{k \in \{0,...,|P|\}, t \in T} -\beta_t + \sum_{i,j} \alpha_{i,j} B_{i,j}^k \tag{16}$$

Such that we search the minimizing feasible path from the root to a fixed but variable terminal and subsequently choose the one path that was minimal with its corresponding terminal.

Next we discuss if the pricing problem is NP-hard. First note that we solve pricing problem for a path from the root to a terminal separately for all terminals. Since there is always one edge to a terminal in a feasible solution, we can equivalently solve the objective $min_k \sum_{i,j} \alpha_{i,j} B_{i,j}^k$ and then add $-\beta_t$ after finding the minimum path. Note by the duality conversion rules that $\alpha_{i,j} \geq 0\ \forall i,j$. The interpretation of the pricing problem is then: Find the optimal column $B_{i,j} \in \mathbb{B}^{|A|}$ such that $min \sum_{i,j} B_{i,j} \alpha_{i,j}$ under the constraint that $B_{i,j}$ encodes a feasible path $p \subset P$ from the root to a terminal $t \in T$, that has the lenght constraint $|p| \leq H$. This is an instance of the constrained cheapest path problem. It can be solved by the following algorithm: For any vertex $v \in V$ we denote the Neighbors $N(v) : \{w \in V : (v,w) \text{ or } (w,v) \in A\}$. Let C(v,k) denote the cost of the cheapest path from the root vertex to the vertex v with a maximum length of k. The cheapest path satisfies the following recurrence relation:

$$C(v, k+1) = \min_{u \in N(v)} (C(u,k) + c((u,v))) \tag{17}$$

where $c((u,v))$ denotes the cost of the arc (u,v). To find the cheapest path from the root to any terminal we start by determining $C(v,1)$ for all $v \in V \setminus \{0\}$ which are just the costs of all the arcs from the root to any other vertex. Next we set $k = 2$ and determine $C(v, 2)$ for all $v \in V \setminus \{0\}$ by the recurrence relation above. We repeat this for all the values of k till we can evaluate $C(t, k = H)$ for all $t \in T$ which then yields the costs of the cheapest paths to all terminals. If we save the generated cheapest paths in every iteration we thereby have an algorithm to generate all cheapest paths to a terminal T with less or equal than H length. Next we encode every path in the column $B^k_{i,j}$ for all terminals $t \in T$ and then take the one that minimizes $-\beta_t + \sum_{i,j} \alpha_{i,j} B^k_{i,j}$ with $B^k_{i,t} = 1$. To determine the cheapest path we need to evaluate in every iteration $k = 1, 2, ..., H$ the expression $C(v, k)$ for all vertices v. The complexity of doing this is $\mathcal{O}(Hn^2)$. Therefore, the pricing problem is tractable and can be solved in polynomial time.

```
[ ]:
```