LINFO2266 - Solutions

November 30, 2021

1 Lagrangian Relaxation

1.1 Set Covering Problem

1. We introduce one Lagrangian multiplier for each constraint to obtain the Lagrangian relaxation:

$$\min \sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} \lambda_j \left(1 - \sum_{\substack{i=1 \\ j \in S_i}}^{m} x_i \right) \tag{1}$$

with $\lambda_j \geq 0, j \in \{1, \ldots, n\}$. Note how we integrated the constraints: for any *feasible* solution of the original problem, we will have

$$1 - \sum_{\substack{i=1 \\ j \in S_i}}^{m} x_i \leq 0 \tag{2}$$

which means a *negative* value will be added to the objective function and that we will thus obtain a *lower bound* on the optimal value of the problem.

For clarity, we rewrite eq. (1) as:

$$\min \sum_{i=1}^{m} \left(c_i - \sum_{j \in S_i} \lambda_j \right) x_i + \sum_{j=1}^{n} \lambda_j \tag{3}$$

For fixed values of the Lagrangian multipliers λ_j, we can obtain a lower bound by solving eq. (3). This problem is very simple to solve if we notice that:

- each x_i is multiplied by a fixed coefficient $\left(c_i - \sum_{j \in S_i} \lambda_j \right)$
- the second term $\left(\sum_{j=1}^{n} \lambda_j \right)$ is constant

In order to minimize the objective function, it is sufficient to select the sets which have a negative coefficient:

$$x_i = \begin{cases} 1 & \text{if } c_i - \sum_{j \in S_i} \lambda_j \leq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

2. By applying exactly what we described above, we get a lower bound value of 4.2.

3. We adapt the subgradient procedure covered in the lectures for this particular problem, and for multiple Lagrange multipliers in algorithm 1.

Algorithm 1 Subgradient procedure for the Set Covering Problem.

$\mathcal{L}^* \leftarrow -\infty, k \leftarrow 1, \mu_0 \leftarrow 1$
$\lambda_{0,j} \leftarrow 0, 1 \leq j \leq n$
$\mathcal{C}^* \leftarrow$ a trivial solution, or none
while $\mu_k \geq \epsilon$ **do**
  Compute cover $\mathcal{C}_k$ with weights $\left(c_i - \sum_{j \in S_i} \lambda_{k,j}\right)$ for $1 \leq i \leq m$
  $\mathcal{L}_k \leftarrow lagrangianValue\,(\mathcal{C}_k)$ (given by eq. (3))
  if $\mathcal{L}_k > \mathcal{L}^*$ **then**
    $\mathcal{L}^* \leftarrow \mathcal{L}_k$
  end if
  if $\mathcal{C}_k$ is feasible and $value\,(\mathcal{C}_k) < value\,(\mathcal{C}^*)$ **then**
    $\mathcal{C}^* \leftarrow \mathcal{C}_k$
  end if
  if $\mathcal{L}^* = value\,(\mathcal{C}^*)$ **then**
    break
  end if
  $\lambda_{k+1,j} \leftarrow \max\left(0, \lambda_{k,j} + \mu_k \left(1 - \sum_{\substack{i=1 \\ j \in S_i}}^{m} x_i\right)\right)$ for $1 \leq j \leq n$
  $\mu_{k+1} \leftarrow \frac{1}{k}$ (or another valid update rule)
  $k \leftarrow k + 1$
end while
return $\mathcal{L}^*$ and $\mathcal{C}^*$

2 Local search

2.1 Strawberry Problem

In this approach, a column represents a single greenhouse placed at a precise location on the field. Formally, each greenhouse is a pattern p defined by a matrix G_p where

$$G_{pij} = \begin{cases} 1 & \text{if greenhouse } p \text{ covers cell } (i,j), \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

for all $1 \leq i \leq A$ and $1 \leq j \leq B$. We suppose that G_{pij} covers a rectangular area of the field.

We associate a variable x_p to each pattern to decide whether it is taken in the solution or not. This allows us to write the objective function and the constraints of the problem:

$$\min \sum_p x_p \left(C_g + \sum_{i=1}^{A} \sum_{j=1}^{B} G_{pij} C_u\right) \tag{6}$$

$$\sum_p x_p G_{pij} \leq 1 \qquad 1 \leq i \leq A, 1 \leq j \leq B \tag{7}$$

$$\sum_p x_p G_{pij} \geq s(i,j) \qquad 1 \leq i \leq A, 1 \leq j \leq B \tag{8}$$

In eq. (6), for each selected greenhouse, we add the fixed cost C_g plus the unit cost C_u for each cell covered by it. Equation (7) states that no two selected greenhouses can both cover a cell. Similarly, eq. (8) ensures that all strawberries are covered by a greenhouse.

Since the number of possible greenhouses is very large, we will start with a small number of them and then introduce new promising patterns, based on intermediate solutions. To initialize the algorithm, a simple idea would be to have one greenhouse covering each cell with a strawberry.

We now formulate the *pricing problem* which will find new interesting columns, if any. A column will be able to improve a current solution if it has a *negative reduced cost.* As in the Simplex algorithm, the goal of the pricing problem is to find the column with the most negative reduced cost.

Given a linear program in matrix form:

$$\min\ c^\top x \tag{9}$$
$$Ax \leq b \tag{10}$$
$$x \geq 0 \tag{11}$$

and its dual linear program:

$$\max\ b^\top y \tag{12}$$
$$A^\top x \geq c \tag{13}$$
$$y \geq 0 \tag{14}$$

the reduced cost vector is given by:

$$\overline{c} = c - A^\top y \tag{15}$$

For a column p of our problem, we obtain the following reduced cost:

$$\begin{aligned}\overline{c_p} &= C_g + \sum_{i=1}^{A}\sum_{j=1}^{B} G_{pij}C_u - \sum_{i=1}^{A}\sum_{j=1}^{B}\alpha_{ij}G_{pij} - \sum_{i=1}^{A}\sum_{j=1}^{B}\beta_{ij}(-G_{pij}) & (16)\\ &= C_g + \sum_{i=1}^{A}\sum_{j=1}^{B} G_{pij}\left(C_u - \alpha_{ij} + \beta_{ij}\right) & (17)\end{aligned}$$

Finally, the pricing problem is given by:

$$\min C_g + \sum_{i=1}^{A}\sum_{j=1}^{B} G_{ij}\left(C_u - \alpha_{ij} + \beta_{ij}\right) \tag{18}$$

with G_{ij} the decision variables and G describing a valid greenhouse. This problem resumes to the Maximum Sum Rectangle problem (try to find how) which can be solved with a well-known dynamic programming algorithm based on Kadane's algorithm.

3 Local search

3.1 Pigment Sequencing Problem

In order to formulate the cost invariants, we use the variables $x_p \in I \cup \{idle\}$ which decide which item is produced at period $p \in [0, p_{max}]$ or if the machine is idle.

If we were to change the value of x_p from i to j and with $i, j \in I \cup \{idle\}$ and $d_i, d_j \geq p$ if $i, j \neq idle$ respectively, the cost of the solution would be increased by a Δ given by:

$$\Delta = \Delta_{stocking} + \Delta_{changeover} \tag{19}$$

where $\Delta_{stocking}$ and $\Delta_{changeover}$ respectively denote the variation of the total stocking and changeover costs.

We first define the individual stocking cost:

$$c_{stocking}(x_p = i) = \begin{cases} 0 & \text{if } i = idle, \\ (d_i - p)h & \text{otherwise.} \end{cases} \tag{20}$$

which allows to formulate $\Delta_{stocking}$ easily:

$$\Delta_{stocking} = c_{stocking}(x_p = j) - c_{stocking}(x_p = i) \tag{21}$$

Similarly, we will express the individual changeover cost. To simplify the coming equations, let us define:

$$q(i, j) = \begin{cases} 0 & \text{if } i = idle \text{ or } j = idle, \\ q^{t_i, t_j} & \text{otherwise.} \end{cases} \tag{22}$$

Let p', p'' respectively denote the previous and next periods where an item is produced (i.e. the machine is not idle). Formally, we write:

$$p' = \max\left(\{\pi \mid \pi \in [0, p-1] \text{ and } x_\pi \neq idle\}\right) \tag{23}$$

$$p'' = \min\left(\{\pi \mid \pi \in [p+1, p_{max}] \text{ and } x_\pi \neq idle\}\right) \tag{24}$$

The items produced at those periods, or the value *idle* if there is no such period, are given by $k = x_{p'}$ and $l = x_{p''}$.

$$c_{changeover}(x_p = i) = \begin{cases} q(k, l) & \text{if } i = idle, \\ q(k, i) + q(i, l) & \text{otherwise.} \end{cases} \tag{25}$$

Finally, we can write:

$$\Delta_{changeover} = c_{changeover}(x_p = j) - c_{changeover}(x_p = i) \tag{26}$$