# 4
# CGI - Column Generation Improvement for Heuristics

Most of the interesting problems are $\mathcal{NP}$-complete. One of the most used ways get solutions to these problems is to use heuristics.

The method described in this chapter (CGI - Column Generation Improvement for Heuristics) is a way to "guide" heuristics of a broad family of problems, so that we can get better results.

In the next section some basic notions of linear programming are presented, since they are necessary for the understanding of the CGI method. Then the family of problems is presented and finally the method itself is presented.

## 4.1 Linear Programming

The linear programming problem consists in optimizing a linear function, under linear constraints.

Basically, it is a problem with the following structure:

$$\begin{aligned} z = \min \quad & c^T x \\ \text{s.t.:} \quad & Ax = b \\ & Bx \leq d \end{aligned}$$



It can be shown that every linear program (an instance of linear programming) can be written in the "standard form" [chvatal1983]:

$$\begin{aligned} z = \min \quad & c^T x \\ \text{s.t.:} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

From now on it will be assumed that all LP's are in the standard form.

### 4.1.1 simplex

The simplex algorithm is one of the most popular algorithms for linear programming (and one of the most important algorithms ever developed).

Now we will make some definitions in order to outline the algorithm:

**Definition 6** *Let $X$ ($|X| = n$) be the set of variables and $m$ be the number of constraints in a linear program. Then, a set $B \subseteq X$ (with $|B| = m$) is called*

*a **basis** if the sub-matrix of $A$ formed by the columns corresponding to $B$ is non-singular.*

*All members $x \in B$ are called **basic variables**.*

*The **basic solution** related to the basis $B$ is the unique solution to the linear program such that all the non-basic variables are set to 0.*

**Definition 7** *Two basis $B$ and $B'$ are **adjacents** if $|B \cap B'| = m - 1$*

The simplex algorithm is possible due to the following two theorems (whose proves can be found at [chvatal1983]):

**Theorem 8** *All linear program with at least one optimal solution has an optimal basic solution.*

**Theorem 9** *Let $B$ be a basis for a linear program. If that program has an optimal solution, then that exists a sequence of basis $B = B_1, \ldots, B_p$ where $B_i$ and $B_{i+1}$ are adjacents, and $B_p$ is an optimal basis.*

*Also, the value associated with basis $B_{i+1}$ is less than or equal to the value associated with basis $B_i$.*

The simplex algorithm "walks" from basis to basis, aiming to improve the solution[1], until the optimum is reached. Since there is a finite number of basis, the optimum is always found.

We will now outline the process to choose the next variable to enter a basis.

Let $B$ be the current basis. We can create a partition of the variable vector in two: $x_b$ and $x_n$, where $x_b$ is the vector of basic variables and $x_n$ is the vector of non-basic variables.

Then, we can rewrite the linear program:

$$\begin{aligned} z = \min \quad & c^T x \\ \text{s.t.:} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

[1] the solution never gets worse, but it can remain at the same value if degeneracy happens. See [chvatal1983]

as the problem:

$$\begin{aligned} z = \min \quad & c_b^T x_b + c_n^T x_n \\ \text{s.t.:} \quad & A_b x_b + A_n x_n = b \\ & x_b \geq 0 \\ & x_n \geq 0 \end{aligned}$$

From $A_b x_b + A_n x_n = b$ we conclude that $x_b = A_b^{-1} b - A_b^{-1} A_n x_n$. Then, the objective function can be rewritten as:

$$\begin{aligned} c_b^T x_b + c_n^T x_n &= c_b^T (A_b^{-1} b - A_b^{-1} A_n x_n) + c_n^T x_n \\ &= c_b^T A_B^{-1} b + (c_n^T - c_b^T A_b^{-1} A_n) x_n \end{aligned}$$

Since we want to minimize this value, $c_b^T A_b^{-1} b$ is constant and $x_n \geq 0$, then we are looking for a direction $x_n \geq 0$ such that the differential in this direction is negative. This is the same as searching for a negative entry in the row vector $c_n^T - c_b^T A_b^{-1} A_n$. If such entry does not exist, the solution with $x_n = 0$ is a local optimum, and since we are optimizing a convex function over a convex region, this is also a global optimum.

The vector $\pi^T = c_b^T A_b^{-1}$ is the solution to the dual of the original linear program, whenever it is feasible for this dual, so $\pi$ will be called the "dual solution" (being feasible or not), and the row vector $c_n^T - \pi^T A_n$ will be called the "reduced price vector".

Although there are already polynomial solutions for the linear programming problem, the simplex algorithm (which is exponential on the worst case) is still used, due to its simplicity and its high performance in practice.

## 4.2 Column Generation

When a linear program has a large number of columns, the execution time can be dominated by the computing of the reduced cost vector (explained in 4.1.1). Sometimes we can find a non-basic variable with negative reduced cost implicitly, without computing every reduced cost.

This is the main idea of the column generation approach. We can formalize the column generation step as solving the following problem:

$$\begin{aligned} \min \quad & c_i - \pi^T A_i \\ \text{s.t.} \quad & i \text{ is a non-basic variable} \end{aligned}$$

Where $A_i$ is the $i$-th column of $A$.

Actually, we don't even need to minimize the problem above, we only need to find a non-basic variable such that $c_i - \pi^T A_i$ is negative, or a proof that it does not exist.

There are several classic problems solved by column generation, such as *cutting stock* and several routing and scheduling problems.

As an example to this technique, consider the *cutting stock* problem, which can be defined as:

You produce rolls of length $L$, and you have several orders $(n_i,\ l_i)$, meaning that you need to produce $n_i$ pieces of roll of length $l_i$ ($l_i \leq L$). You want to deliver all the orders while producing the minimum amount of rolls of length $L$. One way to formulate this problem as a LP is:

Let $\mathcal{J}$ be the set of all possible ways of cutting a roll of length $L$ in rolls of length $l_i$, such that $A_{ij}$ is the number of rolls of length $l_i$ in the cut of "type" $j \in \mathcal{J}$. Clearly $\sum_i l_i A_{ij} \leq L$ for all $j \in \mathcal{J}$.

$$
\begin{aligned}
z = \min \quad & \sum_{j \in \mathcal{J}} x_j & & \\
\text{s.t.:} \quad & \sum_{j \in \mathcal{J}} A_{ij} x_j \geq n_i & \forall (n_i, l_i) \quad & (4.1) \\
& x_j \in \mathbb{Z} & & (4.2) \\
& x_j \geq 0 & &
\end{aligned}
$$

Here the constraints (4.1) make sure that all the orders are delivered [2]

Clearly the number of variables is too big to solve this problem directly using the simplex method, so we will use Column Generation.

In this case, the column generation problem can be written as:

$$
\begin{aligned}
\min \quad & 1 - \sum_i \pi_i A_i \\
\text{s.t.} \quad & \sum_i l_i A_i \leq L \\
& A_i \in \mathbb{Z} \\
& A_i \geq 0
\end{aligned}
$$

Note that $\min 1 - \sum_i \pi_i A_i = 1 - \max \sum_i \pi_i A_i$. Using this observation it is easy to note that solving the column generation problem for the cutting stock is equivalent to the knapsack problem, which is very well solvable in pseudo-polynomial time $O(nL)$ using dynamic programming (here $n$ is the number of pairs $(n_i, l_i)$).

[2] Actually, because of constraints (4.2), the problem is not a linear program, but an *integer linear program*. For the sake of simplicity, we will drop these constraints (considering them is out of the scope of this chapter).

## 4.3 The Linear 0-1 Problem

We will now describe a very broad family of problems, which can be guided by CGI.

**Definition 10 (linear 0-1 problem)** *The linear 0-1 problem is the following problem:*

$$\begin{aligned} \min f(x) \quad & = c^T x \\ s.t.: \quad & x \in \{0,1\}^n \\ & x \in X \end{aligned} \tag{4.3}$$

*Where $x$ is the decision variable (vector) and $X$ is an arbitrary set.*

The problem (4.3) can model a huge variety of problems, such as:

**TSP** let $x_e$ represent if the edge $e$ is part of the solution, and $X$ be the set $\{x|x$ represents a Hamiltonian cycle$\}$.

**UBQP** let $x_{ij}$ represent $y_i y_j$. To do that, set $X = \{x|$there is $y \in \{0,1\}^k$, $x_{ij} = y_i y_j\}$.

**Knapsack** let $x_i = 1$ if and only if object $i$ is in the solution. Also, let $X = \{x|\sum_i w_i x_i \leq W\}$, where $w$ is the weight of each object, and $W$ is the capacity.

Since this is a huge family of problems, the CGI method is very general.

## 4.4 The Proposed Improvement Method

Let $f$, $c$ and $X$ be parts of a linear 0-1 problem $P$ as defined at definition 10. $P$ can be modeled as an LP as:

$$\min \quad f'(y) = \sum_{x \in X} f(x) y_x \tag{4.4}$$

$$\text{s.t.:} \quad \sum_{x \in X} y_x = 1 \tag{4.5}$$

$$\sum_{\substack{x \in X \\ x_i = 1}} y_x \leq 1 \qquad \forall i \in \{1, \ldots, n\} \tag{4.6}$$

$$y_x \leq 1 \qquad \forall x \in X \tag{4.7}$$

$$y_x \geq 0 \qquad \forall x \in X \tag{4.8}$$

The idea is that exactly one $y_x = 1$, and the other will be 0. The $x$ with $y_x = 1$ would be the optimal solution to $P$. In fact, this would be the case even if we drop (4.6). The reason for its inclusion will be clear in a few paragraphs.

**Theorem 11** *There is am optimal solution for* (4.4) *where exactly one* $y_x = 1$ *and all the other are* 0*.*

*Proof.* Let $y^*$ be an optimal solution for (4.4). Let $Y = \{x \in X | y_x > 0\}$.

We will first prove that $\forall x_1, x_2 \in Y$ we have that $f(x_1) = f(x_2)$.

Suppose there are $x_1, x_2 \in Y$ such that $f(x_1) < f(x_2)$, and let $y'$ be:

$$y'_x = \begin{cases} y_x & \text{if } x \notin \{x_1, x_2\} \\ 0 & \text{if } x = x_2 \\ y_{x_1} + y_{x_2} & \text{if } x = x_1 \end{cases}$$

y' is obviously a feasible solution, and $f'(y') = f'(y) + f(x_1)y_{x_2} - f(x_2)y_{x_2} = f'(y) + y_{x_2}(f(x_1) - f(x_2)) < f'(y)$, so $y$ is not optimal, which is a contradiction.

So, if $\forall x_1, x_2 \in Y$ we have that $f(x_1) = f(x_2)$, then the solution $y^x$ which has $y^x_x = 1$ is optimal for every $x \in Y$.

∎

Since the set $X$ is usually very big, a column generation approach is used (as explained at 4.2).

The column generation subproblem would be formulated as:

$$\begin{aligned} \min \quad & f(x) - \pi_0 - \sum_{i|x_i=1} \pi_i \\ & = f(x) - \pi_0 - \pi^T x \\ & = c^T x - \pi^T x - \pi_0 \\ & = (c - \pi)^T x - \pi_0 \\ \text{s.t.:} \quad & x \in X \end{aligned}$$

That is, given the dual solution $\pi$, the problem of finding a nonbasic variable with negative reduced cost is a linear 0-1 problem, in the same set as the original problem, but with a different linear function. All that we need is a solution to the modified problem whose value is less than $\pi_0$.

In the case of max cut, for instance, this would be another max cut over the same graph, but with edge weight $c - \pi$. We can stop as soon as we get to a cut whose value is less than a given threshold $\pi_0$.

The purpose of CGI would be to weight the "important" variables of the linear 0-1 problem, so that we can guide the heuristics.

## 4.5
## One Last Problem to Solve

There is a problem that was not discussed yet: $\pi' = (f'(y^*), 0, \ldots, 0)$ is always a solution to the dual of the problem 4.4.

The problem with $\pi'$ is that it does not change the linear function of the linear 0-1 problem, all it does is to ask: "Is there a solution to the original problem which is better then the best solution found so far?". There is no need for such a framework just to give us this simple conclusion: "to improve your solution you must improve your solution".

All popular linear solvers (free or commercial) are smart enough to give this solution to the dual problem, so we are in trouble. There is a whole polytope of solutions for the dual problem, and we could try to find a solution $\pi$ which is far from $\pi'$. This is easy since we already know the value of the dual problem, namely, it is the same value of the best solution found so far. Let this value be called $z^*$.

We can then solve the linear problem whose constraints are the same as the constraints of the dual problem, plus $\sum_i \pi_i = z^*$, so that we are optimizing over the optimal polytope. If we optimize for some objective function $d^T\pi$ such that $d_0 = 0$, we will get a dual solution $\pi$ which is the farthest solution from $\pi'$ in the direction $d$. We solve this same problem for some random $d$ (satisfying $d_0 = 0$), and take the farthest to $\pi'$.