Pseudo Code for Repair Algorithm

Arthur Feng

March 25, 2019

1 Introduction

Definition 1.1. (Mixed Integer Programming)
In this report, we denote the $Mixed\,Integer\,programming$ (MIP) problem in the follow form

$$\text{(MIP)} \quad \min\{\, c^T x : Ax \le b, x \in \mathbb{Z}^l \,\text{x}\, \mathbb{R}^k\}$$

where $m, n \in \mathbb{N}$, and $n = l + k$, A is a matrix for $A \in \mathbb{R}^{m\,\text{x}\,n}$, and the vector $b \in \mathbb{R}^m$, the vector $c \in \mathbb{R}^n$.

S is a set of $feasible\,solution$ if S satisfy all the constraints in the problem. If $s \in S$, s is called a feasible solution to the problem $\mathcal{Z}_{MIP}$. The MIP is called infeasible when S is empty, other wise, the MIP is called feasible. The feasible solution s^* is called optimal when $s^* \in S$ and $c^T x_{s^*} \le c^T x_s$ for $\forall s \in S$.

Definition 1.2. (Linear Programming and LP-relaxation)
With the definition of MIP in 1.1, there is a special case of MIP when all variables are continuous, which is called $Linear Programming$ (LP) problem

$$\mathcal{Z}_{LP} = \min\{\, c^T x : Ax \le b, x \in \mathbb{R}^k\}$$

A LP can also be obtained by removing all integrity constraints: $x_i \in x$ where $i \in n \setminus l$, this is called LP-*relaxation*. LP-*relaxation* is the foundation of LP-based branch-and-bound technology. As the searching space is increase by removing integrity restrictions, the optimal solution in MIP could not better than LP-*relaxation*, which is $s^*_{MIP} \ge s^*_{LP}$. This means the optimal solution found in LP could provide a lower or prime bound for MIP.

Definition 1.3. (Binary Programs)
In the definition 1.1, When all variables are integer

$$\mathcal{Z}_{IP} = \min\{\, c^T x : Ax \le b, x \in \mathbb{Z}^l\}$$

The problem is call pure Integer Programming (IP) problem. And if we denote $\mathbb{B}$ be a set of binary values where $\mathbb{B} = \{0, 1\}$ for $\forall x \in \mathbb{B}^l$, then we call this problem a Binary Programming problem. This report we focus on Binary Programs.
Question: the two terminologies $Binary programming problem$ and $pseudo - Boolean problem$ is same? or not?

2 Input

- A problem $\mathcal{P}^0 = min\{c^T x \,|\, Ax \geq b,\, x_i \in \mathbb{B}^n, \mathbb{B} = \{0, 1\}\}$ with n binary variables x, objective function $O^0 = c^T x$ (min), constraint set C^0 with an optimal solution s^0.
- A problem $\mathcal{P}^1$ with n binary variables x, objective function $O^1 = O^0 = c^T x$ (min), constraint set C^1, such that $C^0 \subsetneq C^1$.

3 Output

- An optimal solution s^1 to $\mathcal{P}^1$.

4 Pseudo Code

Algorithm 1: Solving Problem with Reoptimization

```
Data: $r, \mathcal{P}^0, s^0$ and $\mathcal{P}^1$ where $C^0 \subsetneq C^1$ , $O^0 = O^1$
Result: optimal solution $s^1$ to $\mathcal{P}^1$
begin
    if $s^0$ is feasible to $\mathcal{P}^1$ then
        return $s^0$
    else
        $C^+ \longleftarrow C^1 - C^0$
        for $c \in C^+$ do
            add $v_i$ to the set $V$ where it's coefficient $c_i \neq 0$
        end
        $y^0 = c^T x$ for $x \in s^0$
        $y^p = c^T x$ for $x \in V^+$
        $\Delta y = |Y^0 - Y^p|$
        if $\Delta y / y^0 \geq r$ then
            solve $\mathcal{P}^1$ from scratch
        else
            solve it with reoptimization
            $p \longleftarrow (|y^0| + |y^p|)$
            $O^p \longleftarrow p * \sum x$ for $x \in V^+$
            $\mathcal{P}^*_0 \longleftarrow O^p + C1$
            while $p \geq 0$ do
                $p = p//1.2$
                $\mathcal{P}^*_{i+1} \longleftarrow$ update $O^p$
                if $i == 0$ then
                    input $s^i$ to find the first feasible solution
                        question: do we need to fix $s^0$ ?
                end
                $s^*_i \longleftarrow$ reoptimize the sub-problem in SCIP
                if $s^*_i$ don't change in n iterations then
                    $s^1 \longleftarrow s^*_i$
                    return $s_1$
                else if $p = 0$ then
                    $s^1 \longleftarrow s^*_i$
                    return $s_1$
            end
        end
    end
end
```