Constraint propagation and Backtracking

June 1, 2015

1 Definitions

Each constraint i of an Integer Program $\mathcal{P}$ can be written as one or two constraints in the form:

$$\sum_{j \in N} a_{ij} x_j \leq b_i$$

where N is the index set of variables.

Considering that $N_i^+ = \{j \in N : a_{ij} > 0\}$, $N_i^- = \{j \in N : a_{ij} < 0\}$ and l_j and u_j are the lower and upper bounds on x_j , respectively, then the upper bound U_{ij} for the left-hand side of constraint i can be determined as:

$$U_{i\hat{j}} = b_i - \sum_{j \in N_i^+, j \neq \hat{j}} a_{ij} l_j + \sum_{j \in N_i^-, j \neq \hat{j}} |a_{ij}| \, u_j$$

For each variable and constraint there may be a an implication which is computed in function $evaluateBound(\mathcal{P}, l, u, i, j)$ where i is a constraint index and j is a variable index. Possible results for this function are: `CONFLICT`, if a conflict has been detected, `IMPLICATION` if a set of one or more implications was found and `NO_IMPLICATION` if no implication could be obtained.

Algorithm 1: Constraint Propagation

```
Input: P, l, u, C
Output: (status, F)
status ← NO_IMPLICATION; F ← ∅ ;
repeat
    new_implication = false;
    forall constraint c in C do
        forall variable j not fixed in c do
            (status, bound) ← evaluateBound (P, l, u, c, j);
            if ∃ (j, b) ∈ F : b ≠ bound then
                return (CONFLICT, F);
            else
                if status = FIX then
                    F ← F ∪ {(j, bound)};
                    new_implication = true;
                else if status = CONFLICT then
                    return (CONFLICT, F);
    if F = ∅ then
        return (NO_IMPLICATION, ∅);
    forall (j, bound) ∈ F do
        l_j = u_j = bound;
        forall c ∈ C such that a_cj ≠ 0 do
            if c has one or more unfixed variables then
                C ← C ∪ {c};
    forall c in C do
        if all variables in c are fixed then
            if c is unfeasible then
                return CONFLICT;
            C ← C \ {c};
until (C ≠ 0) and (new_implication = true) ;
return (status, F);
```

Algorithm 2: evaluateBound

Input: $\mathcal{P}, l, u, i, j$
Output: $(status, bound)$

$U_{ij} \leftarrow computeU(\mathcal{P}, l, u, i, j)$;
$status \leftarrow$ `NO_IMPLICATION`; $bound \leftarrow$ `NIL`;
if $j \in N_i^+$ **then**
  if $U_{ij} < 0$ **then**
    $status \leftarrow$ `CONFLICT`;
  else if $a_{ij} > U_{ij}$ **then**
    $bound \leftarrow 0$;
    $status \leftarrow$ `FIX`
if $j \in N_i^-$ **then**
  if $a_{ij} > U_{ij}$ **then**
    $status \leftarrow$ `CONFLICT`;
  else if $U_{ij} < 0$ **then**
    $bound \leftarrow 1$;
    $status \leftarrow$ `FIX`;
return $(status, bound)$;

Algorithm 3: Backtrack

Input: *LP instance*, s *Solution*, l, u, v *value*, j *variable*

$l_j = u_j \leftarrow v$;
if *all variable are fixed* **then**
    if *s is feasible* **then**
        if $f(s)$ *is better* $f(s^*)$ **then**
            $s^* \leftarrow s$;
            return;
    else
        return;
$C \longleftarrow$ *all constraints of j which have one or more unfixed variables*;
$(status, F) =$ Constraint propagation (lp, l, u, C);
if $status =$ `CONFLICT` **then**
    return;
else if $status =$ `FIX` **then**
    forall $(j, bound) \in F$ **do**
        $l_j = u_j = bound$;
if *all variable are fixed* **then**
    if $f(s)$ *is better* $f(s^*)$ **then**
        $s^* \leftarrow s$;
        return;
select one j' *unfixed*;
BackTrack $(lp, s, fix, lower, upper, 1, j')$;
BackTrack $(lp, s, fix, lower, upper, 0, j')$;