Optimization (Exercices)

1 Solving COP

Let P be the following COP instance:

```
from pycsp3 import *

x = Var(0, 1, 2)
y = Var(0, 1, 2)
z = Var(0, 1, 2)

satisfy(
    x > y,
    y != z
)

maximize(
    x + y + z
)
```

Assuming that AC is maintained during search (while considering a non-binary branching scheme), and *lexico* is used as variable ordering heuristic, draw the search tree that is built when solving P:

- with values selected in increasing order;
- with values selected in decreasing order.

2 Depth-First Branch and Bound

Depth-first branch and bound (DFBB) performs a depth-first traversal of the search tree, when solving an optimization (minimization) problem. We have that:

- at any time, the upper bound ub is known, which is the maximum cost (-1) that we are willing to accept;
- at any node corresponding to an instantiation I, the lower bound $lb(I)$ is an underestimation of the cost of any complete assignment (extending I) below that node.

When $ub \leq lb(I)$, the subtree rooted by I can be pruned because it contains no solution with a cost lower than ub. If it finds a complete assignment (i.e., we have $|I| = n$, with n being the number of variables in the problem) with a cost lower than ub, this cost becomes the new ub; after exhausting the tree, ub is the optimal cost. The temporal complexity of DFBB is $O(d^n)$, while its spatial complexity is $O(nd)$.

Give the pseudo-code of a recursive algorithm that implements DFBB. The initial call is dfbb($\emptyset$).

3 Solving MaxCSP

We assume binary constraints only, and we are interested in finding the instantiation that satisfies the higher number of constraints (MaxCSP): the cost of a violated constraint is 1 whereas it is 0 if the constraint is satisfied. We propose to use DFBB for solving MaxCSP, with two different ways of computing $lb(I)$ where I is the current instantiation.

- $lb_1(I)$ is the sum of the costs of all constraints covered by the current instantiation (i.e., the constraints only involving fixed variables)
- $lb_2(I)$ is $lb_1(I)$ plus the sum of the minimum possible costs of all constraints almost covered by the current instantiation (i.e., the constraints involving a fixed variable and an unfixed variable)

Let P be the following CN:

- $vars(P) = \{shoes, shirt, slacks\}$ with
  - $dom(shoes) = \{cordovans, sneakers\}$
  - $dom(slacks) = \{jeans, dress, skirt\}$
  - $dom(shirt) = \{green, white\}$
- $ctrs(P) = \{$

  $\langle shoes, slacks \rangle \in \{(cordovans, skirt), (sneakers, jeans)\},$

  $\langle shoes, shirt \rangle \in \{(cordovans, white)\},$

  $\langle slacks, shirt \rangle \in \{(jeans, white), (dress, white), (skirt, green)\}$

  $\}$

While considering the static variable ordering $shoes < slacks < shirt$ and a lexicographic value ordering,

1. draw the search tree built by DFBB for MaxCSP solving of P, when using lb_1,
2. draw the search tree built by DFBB for MaxCSP solving of P, when using lb_2.