# Multi-Stage Constraint Programming

Thi-Van-Anh Nguyen[1], Arnaud Lallouet[1] and Youssef Hamadi[2]

[1] GREYC - University of Caen Lower Normandy - France

[2] Microsoft Research Cambridge - United Kingdom

## Abstract

Multi-agent decision under uncertainty as encountered in sustainable development applications has raised new challenges in optimization. Not only is needed to find optimal value for a static problem but also for situations in which several agents have their own goals and interact by sharing some resources. We propose to address this latter category of problem by introducing a multi-agent multi-level optimization language based on quantified programming. Three aspects will be addressed: modelling, solving, and applications.

## CSP: Constraint Satisfaction Problem

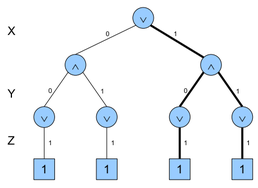
Constraint Programming is a way of ***stating*** and ***solving*** problems using variables and constraints.
A CSP is built out of 3 parts:

- V: a set of variable
- D: a set of domains
- C: a set of constraints

**Example**: $X \in \{1,2,3\}, Y \in \{3,4\}, Z \in \{4,5,6\}. X < Y \wedge X + Y = Z \wedge Z \neq 3X$. X = 1, Y = 4, Z = 5 is a valid solution while X = 2, Y = 4, Z = 6 is not.
**Application:** Scheduling, planning, allocation, optimization,...
**Technique:** Search and Propagation.

## QCSP: Quantified Constraint Satisfaction Problem

QCSP = CSP + $\forall$
A solution = a winning strategy
**Example:** $\exists X \in \{0,1\}, \forall Y \in \{0,1\}, \exists Z \in \{1\} \ . \ X \vee Y = Z$




## $QCSP^+$: QCSP using restricted quantification

### $QCSP^+$ in logic

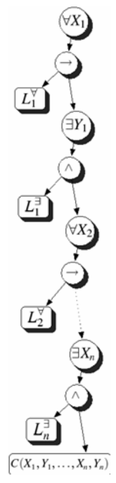
- Syntax:

$QCSP^+ ::==$
$\exists X.CSP \wedge QCSP^+ \mid$
$\forall Y.CSP \rightarrow QCSP^+ \mid$
CSP

- Using [ ] to model "such that":

$\exists X_1[CSP]$
$\forall X_2[CSP]$
$\exists X_3[CSP]...$
...CSP

- [ ] expands to $\wedge$ in existential blocks and to $\rightarrow$ in universal.



**Application:** A friendly modelling tool for problems under uncertainty or adversary.

**Example:** Nim-Black-Jack game modelled by $QCSP^+$

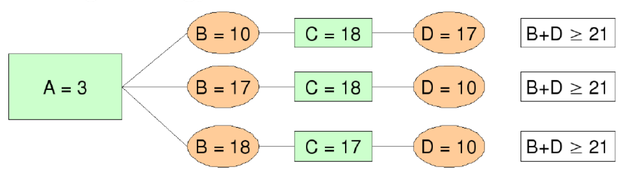
***Rules of the game:***

- Adding contextual rules: 3 10 17 18
- Each player takes a number in turn. The goal is to make a sum greater than the one of the opponent but less than 21
- It is not possible to take the same number twice.

***Model in*** $QCSP^+$ :

$\exists A \in P$
$\forall B \in P(B \neq A)$
$\exists C \in P(C \neq A \wedge C \neq B \wedge A + C < 21)$
$\forall D \in P(D \neq A \wedge D \neq B \wedge D \neq C \wedge B + D < 21)$
$B + D < A + C$

The game has a winning strategy with A = 3.




## Proof number search on $QCSP^+$

Proof number search is a best-first AND/OR search tree algorithm

**Algorithm 1** *PNS* algorithm on $QCSP^+$

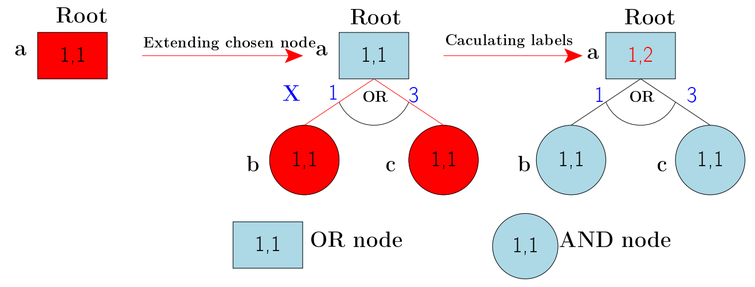
**Input:**
Root node (pn = 1, dn = 1)
**Output:**
True or false

```
while (pn(root) ≠ 0 AND dn(root) ≠ 0) do
    currentNode = chooseNode(root)
    extendNode(currentNode,QCSP+)
    calculateLabel(currentNode)
end while
if pn(root) = 0 then
    return true
end if
if dn(root) = 0 then
    return false
end if
```

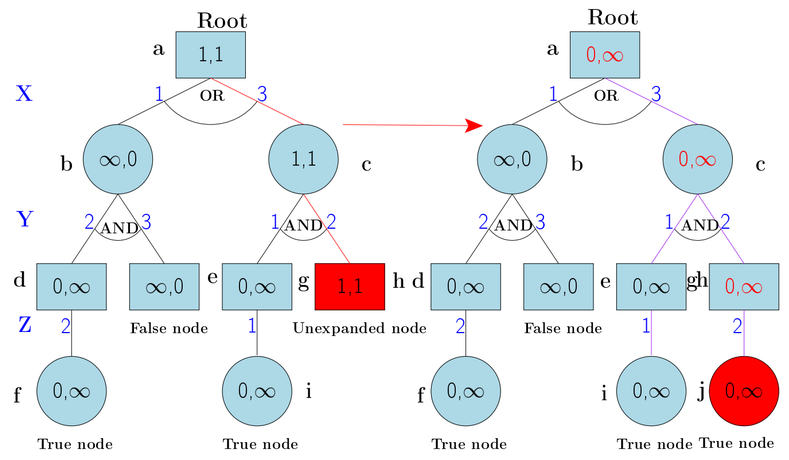
**Example:** $D(X) = D(Y) = \{1,2,3\}, D(Z) = \{1,2\}$

$\exists X(X \neq 2)$
$\forall Y(X \neq Y)$
$\exists Z(Y = Z)$
$Z < 3$

The first loop of PNS algorithm

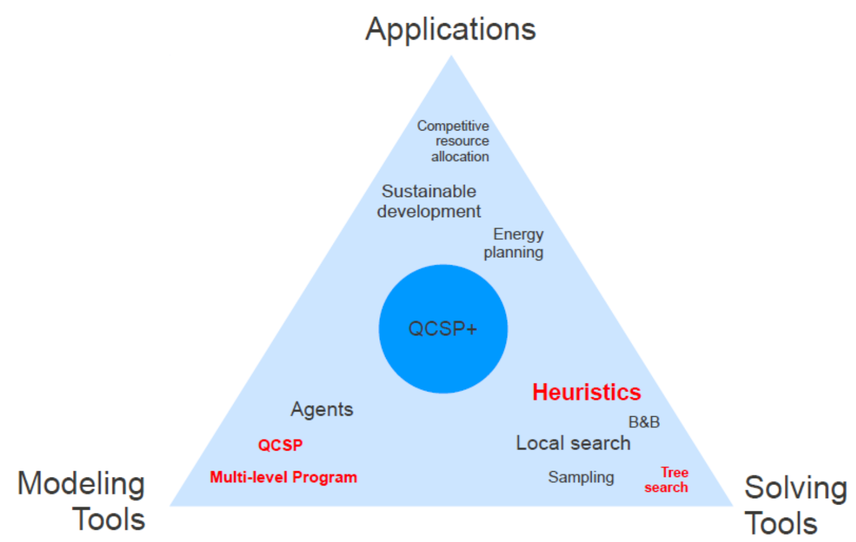



The last loop of PNS algorithm




The node names are mentioned by their existing position. We have a winning strategy noted by the violet lines.

## Conclusion




## Intended outcomes

- Local search
- Multi-agent
- Sampling