QCSP guidé par Monte Carlo

J-M. Chantrein V. Barichard I. Stephan

LERIA

13 juin 2014

JIAF 2014

Prérequis

QCSP : *Quantified Constraint Satisfaction Problem*

Les problèmes de satisfaction de contraintes quantifiés (QCSP) sont une extension des problèmes de satisfaction de contraintes (CSP).
Les variables d'un QCSP peuvent être quantifiées :

- existentiellement ($\exists$)

Prérequis

QCSP : *Quantified Constraint Satisfaction Problem*

Les problèmes de satisfaction de contraintes quantifiés (QCSP) sont une extension des problèmes de satisfaction de contraintes (CSP).
Les variables d'un QCSP peuvent être quantifiées :

- existentiellement ($\exists$)
- universellement ($\forall$)

Prérequis

QCSP : *Quantified Constraint Satisfaction Problem*

Les problèmes de satisfaction de contraintes quantifiés (QCSP) sont une extension des problèmes de satisfaction de contraintes (CSP).
Les variables d'un QCSP peuvent être quantifiées :

- existentiellement ($\exists$)
- universellement ($\forall$)

$\Rightarrow$ permet de modéliser de manière concise des problèmes qui ne peuvent l'être en CSP.

Prérequis

QCSP : *Quantified Constraint Satisfaction Problem*

Les problèmes de satisfaction de contraintes quantifiés (QCSP) sont une extension des problèmes de satisfaction de contraintes (CSP).
Les variables d'un QCSP peuvent être quantifiées :

- existentiellement ($\exists$)
- universellement ($\forall$)

$\Rightarrow$ permet de modéliser de manière concise des problèmes qui ne peuvent l'être en CSP.

- QuaCode est un solveur de QCSP utilisant la bibliothèque CSP Gecode et développé au LERIA.

Exemple

Considérons le QCSP suivant :

$$\exists x \forall y \exists z (x + z \leq y)$$

avec $x, y, z \in \{2, 1, 0\}$

Exemple

Considérons le QCSP suivant :

$$\exists x \forall y \exists z (x + z \leq y)$$

avec $x, y, z \in \{2, 1, 0\}$

- Pour des raisons pédagogiques, le parcours des valeurs des domaines se fera dans l'ordre décroissant.

Exemple

Considérons le QCSP suivant :

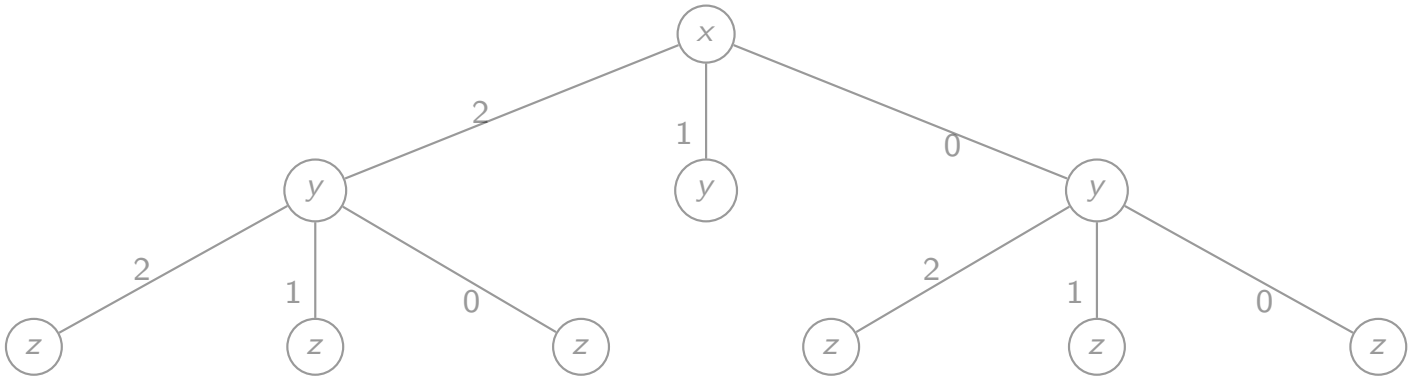
$$\exists x \forall y \exists z (x + z \leq y)$$

avec $x, y, z \in \{2, 1, 0\}$

- Pour des raisons pédagogiques, le parcours des valeurs des domaines se fera dans l'ordre décroissant.
- Observons la résolution de ce QCSP.

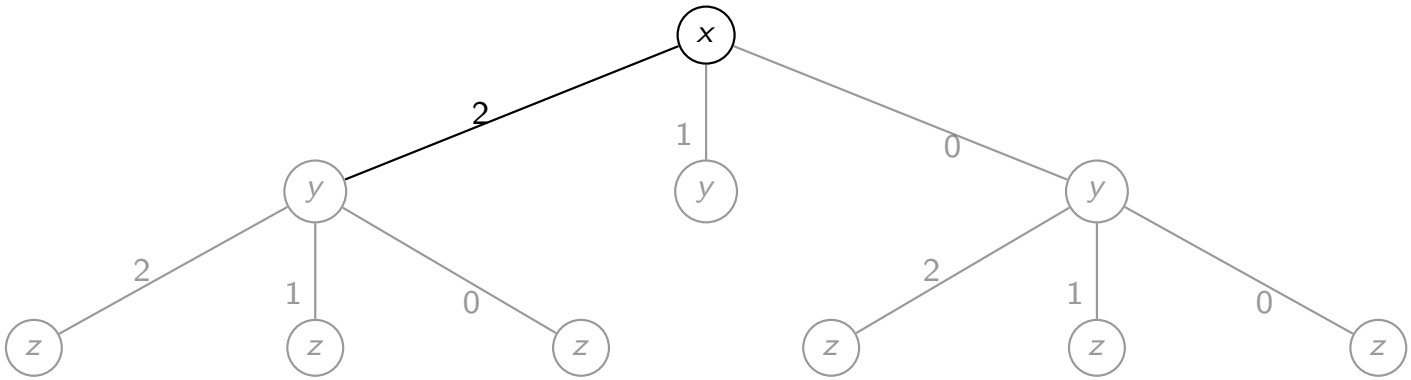
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



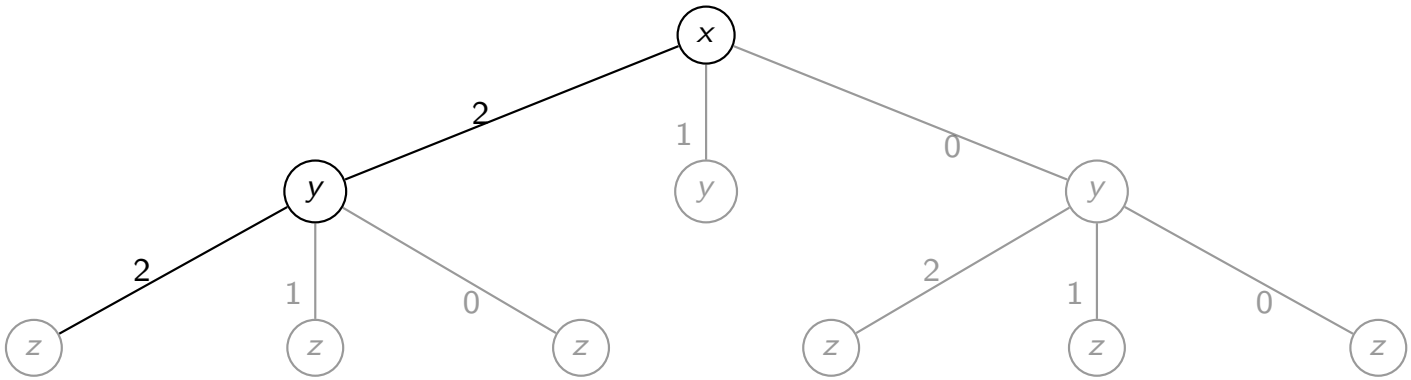
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



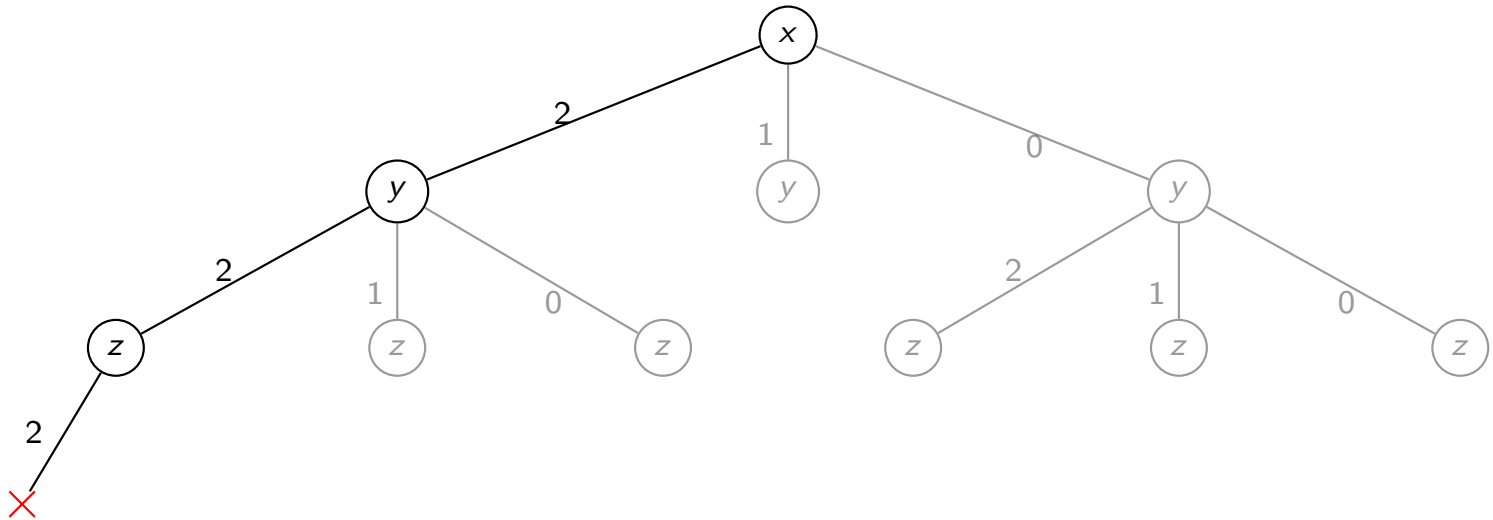
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



Résolution d'un QCSP

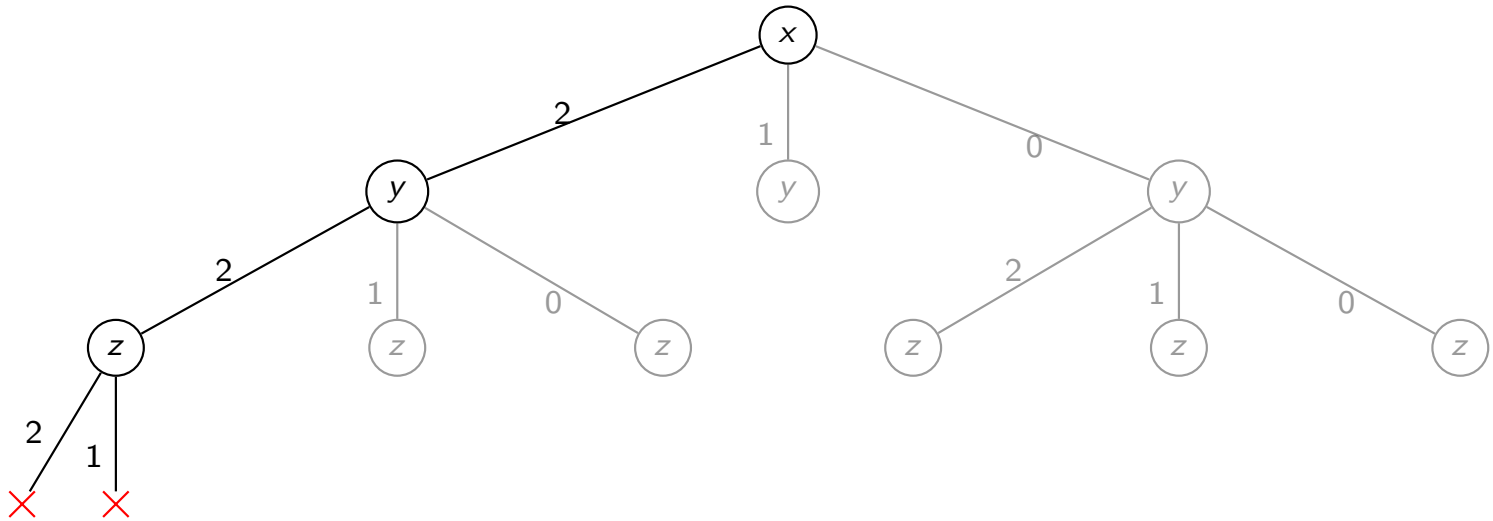
$$\exists x \forall y \exists z (x + z \leq y)$$



$2 + 2 > 2$

Résolution d'un QCSP

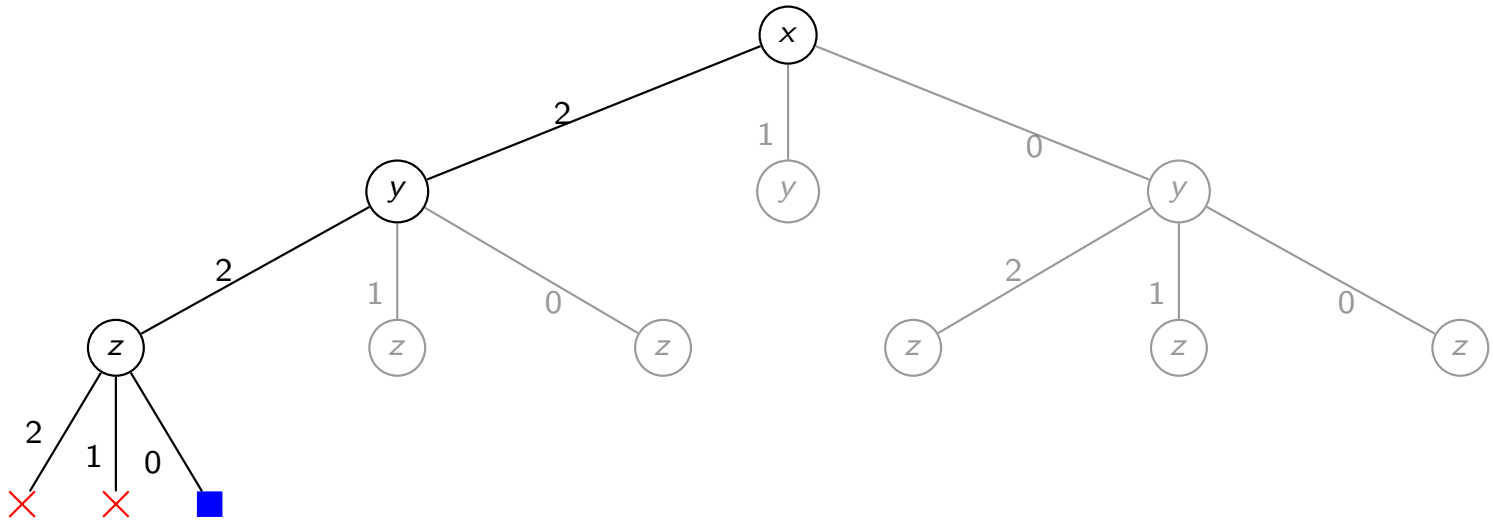
$$\exists x \forall y \exists z (x + z \leq y)$$



$2 + 1 > 2$

Résolution d'un QCSP

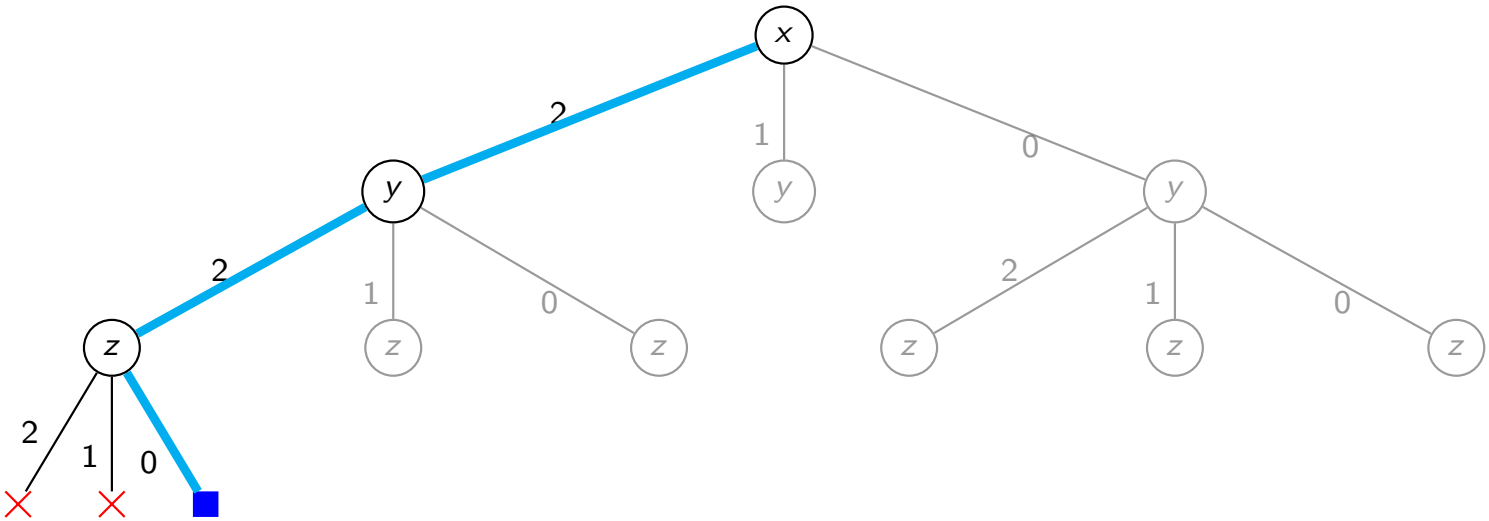
$$\exists x \forall y \exists z (x + z \leq y)$$



$$2 + 0 \leq 2$$

Résolution d'un QCSP

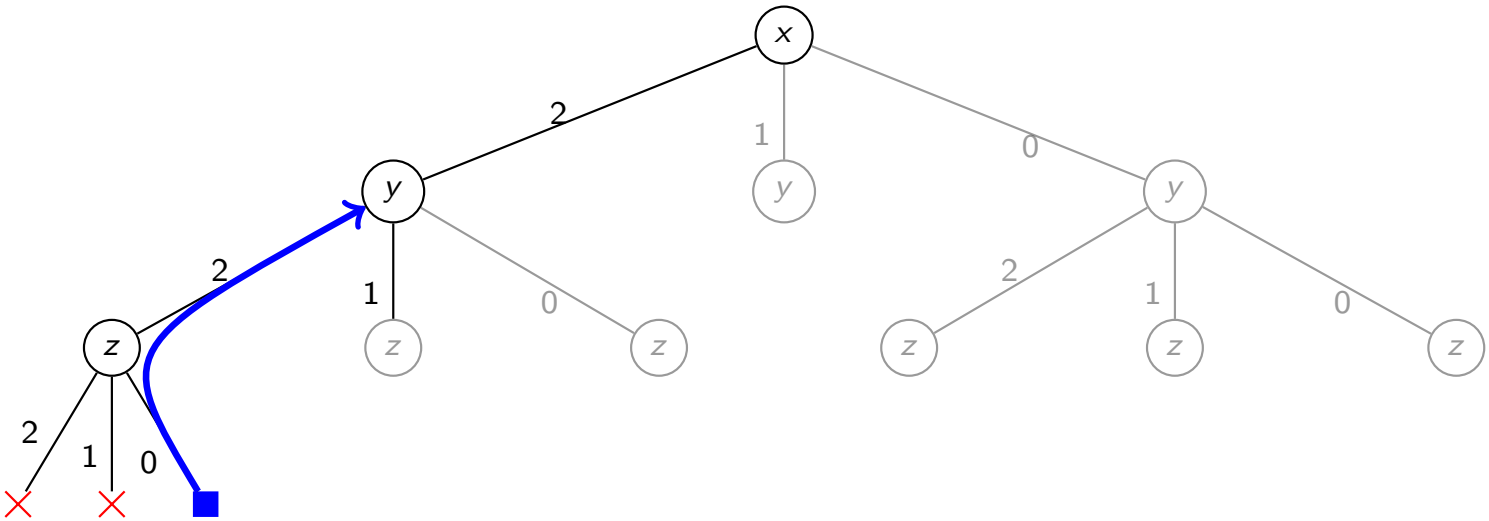
$$\exists x \forall y \exists z (x + z \leq y)$$



Scénario gagnant : $(x = 2)$, $(y = 2)$, $(z = 0)$

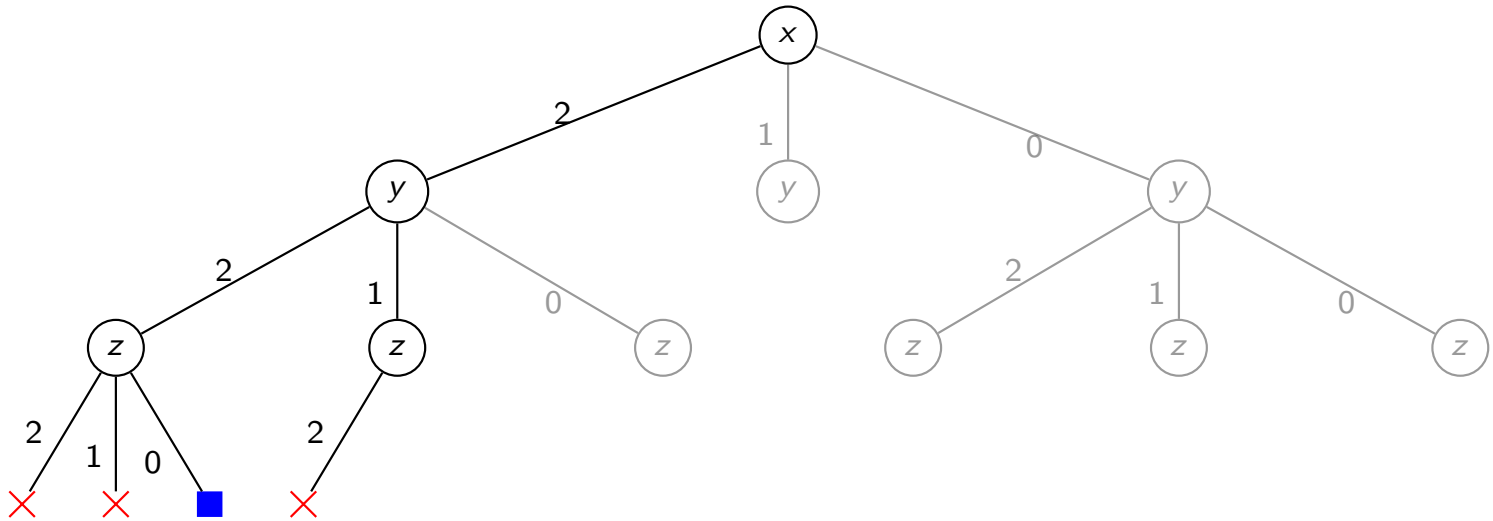
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



Résolution d'un QCSP

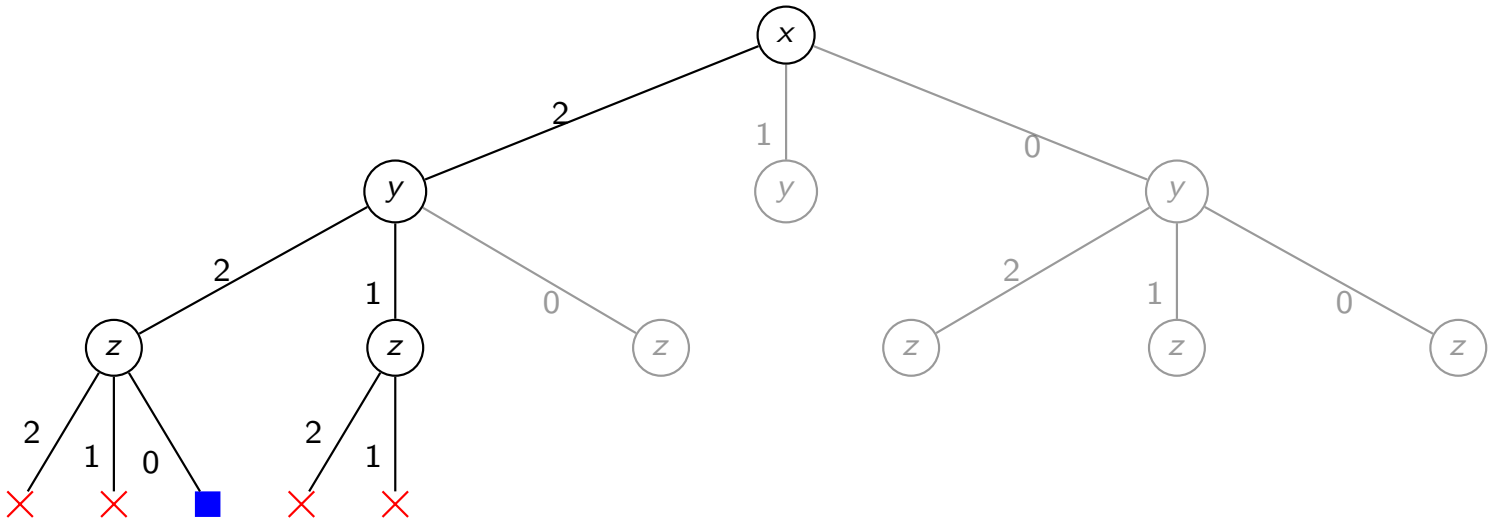
$$\exists x \forall y \exists z (x + z \leq y)$$



$2 + 2 > 1$

Résolution d'un QCSP

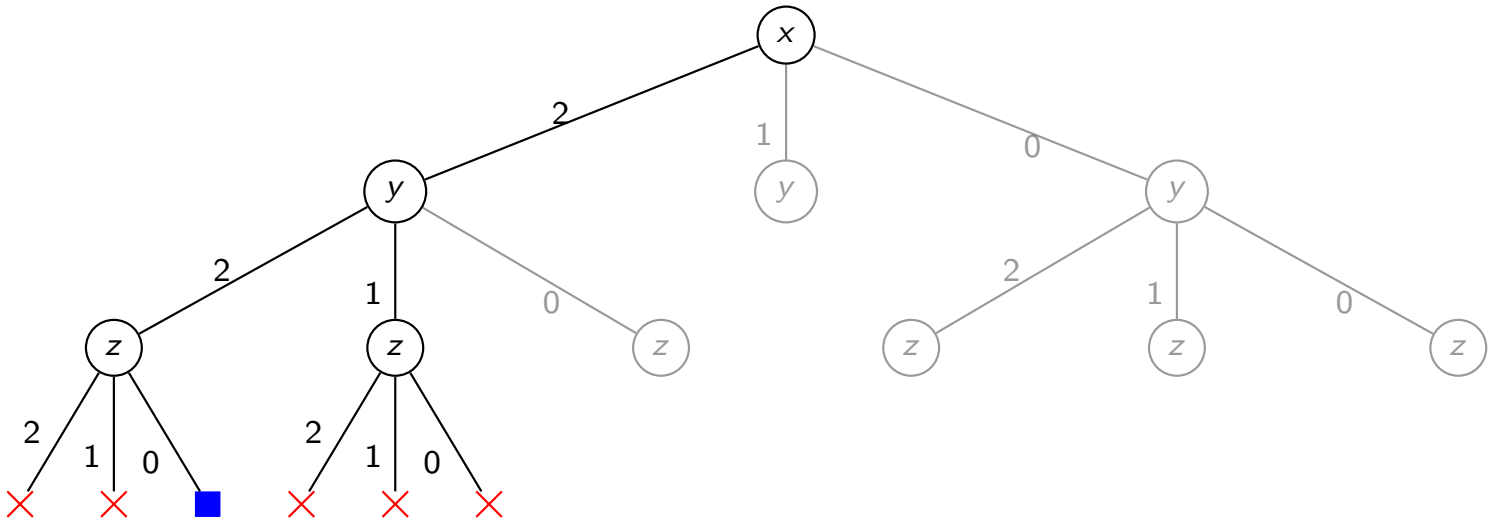
$$\exists x \forall y \exists z (x + z \leq y)$$



$2 + 1 > 1$

Résolution d'un QCSP

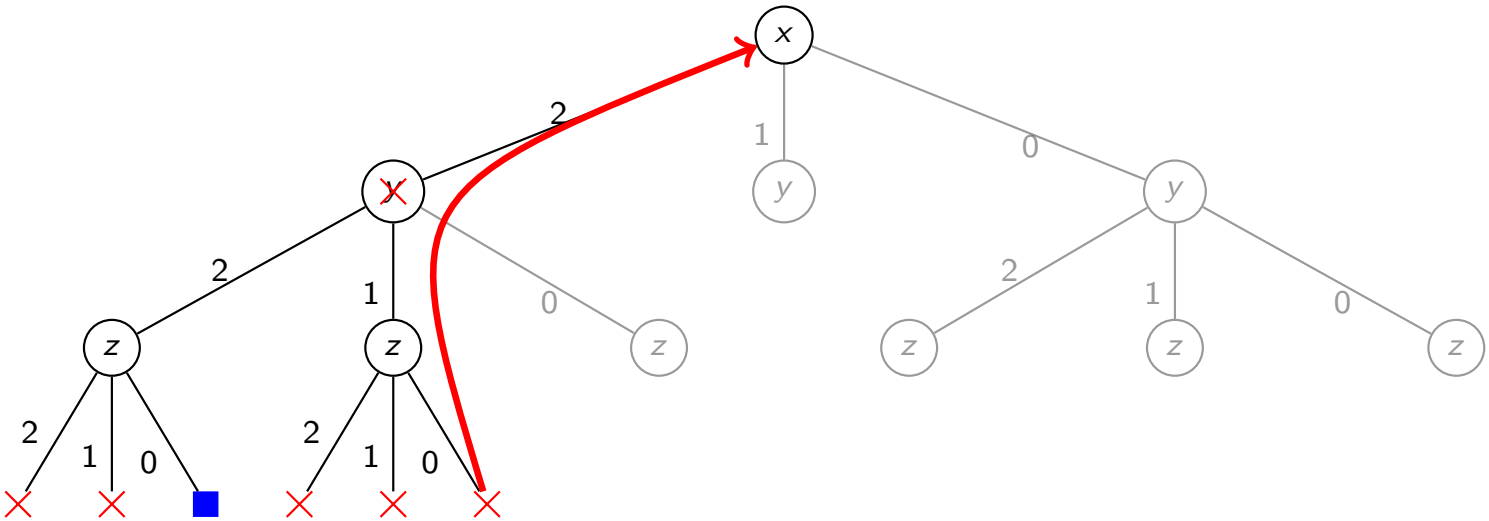
$$\exists x \forall y \exists z (x + z \leq y)$$



$2 + 0 > 1$

Résolution d'un QCSP

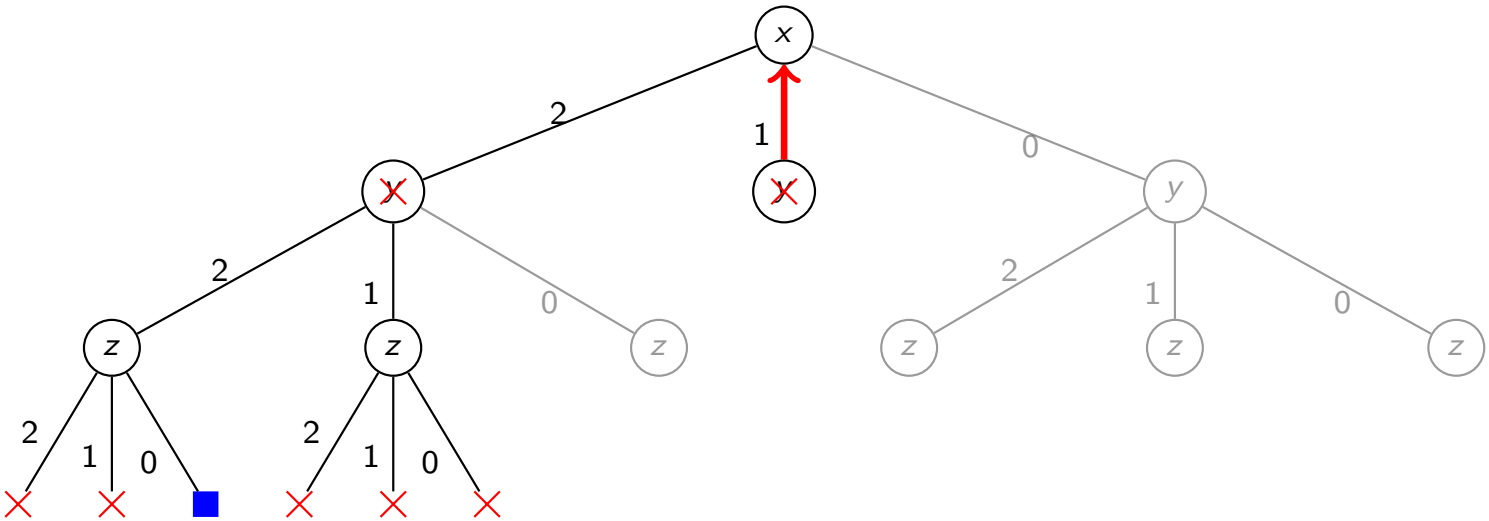
$$\exists x \forall y \exists z (x + z \leq y)$$



retour en arrière

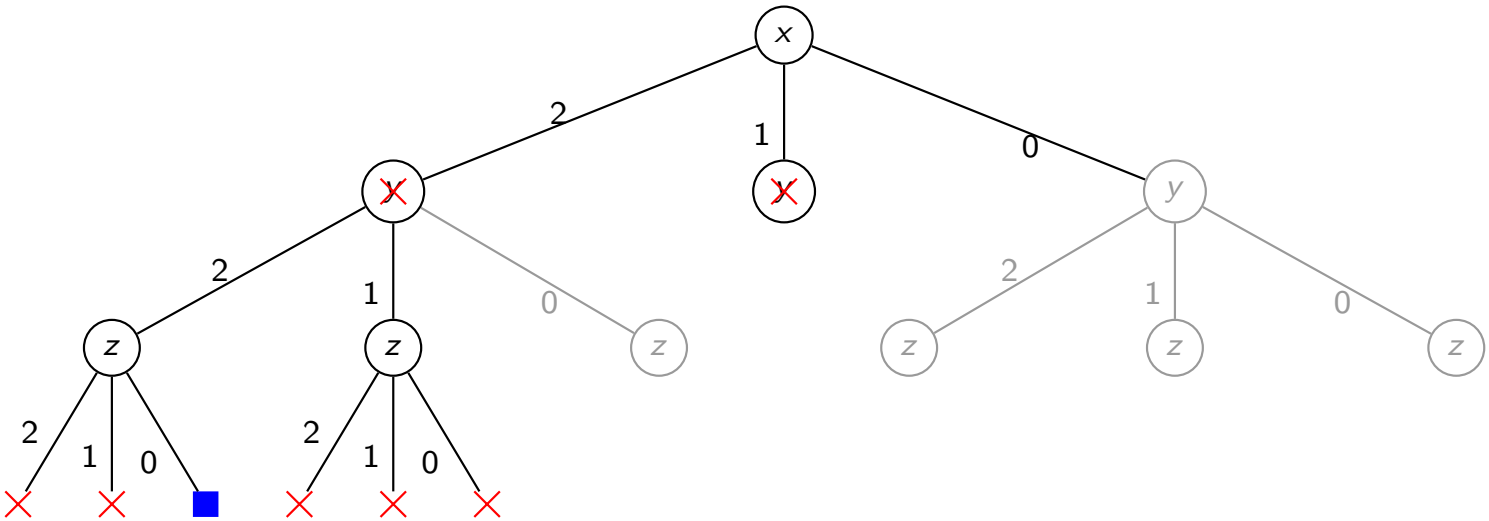
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



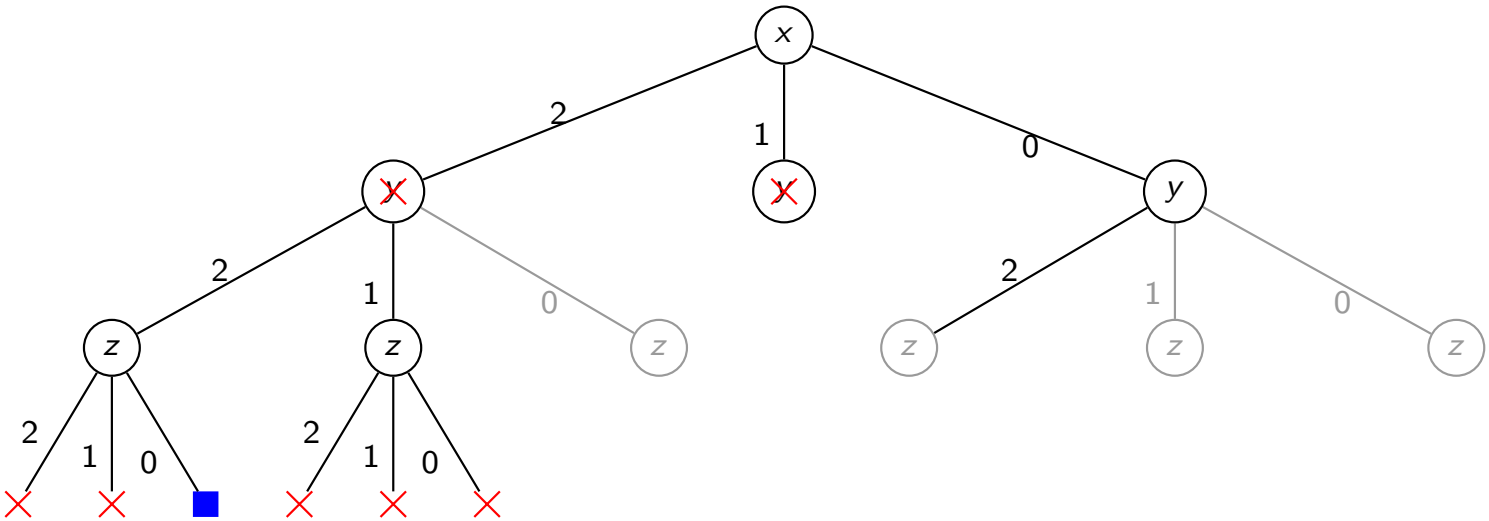
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



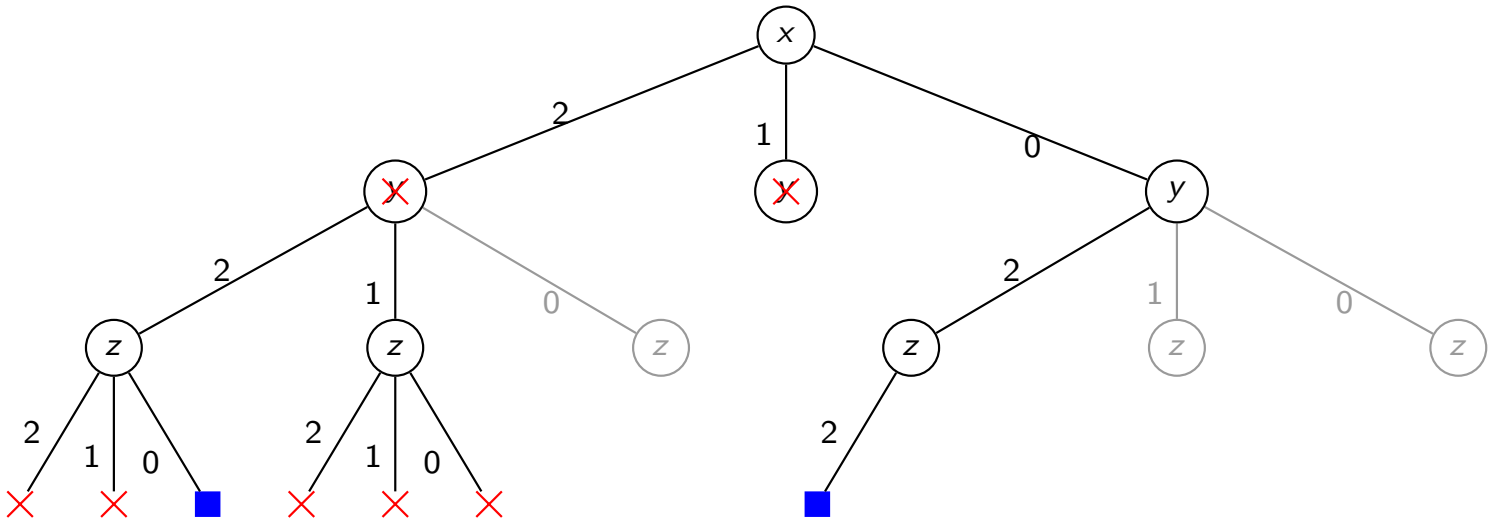
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



Résolution d'un QCSP

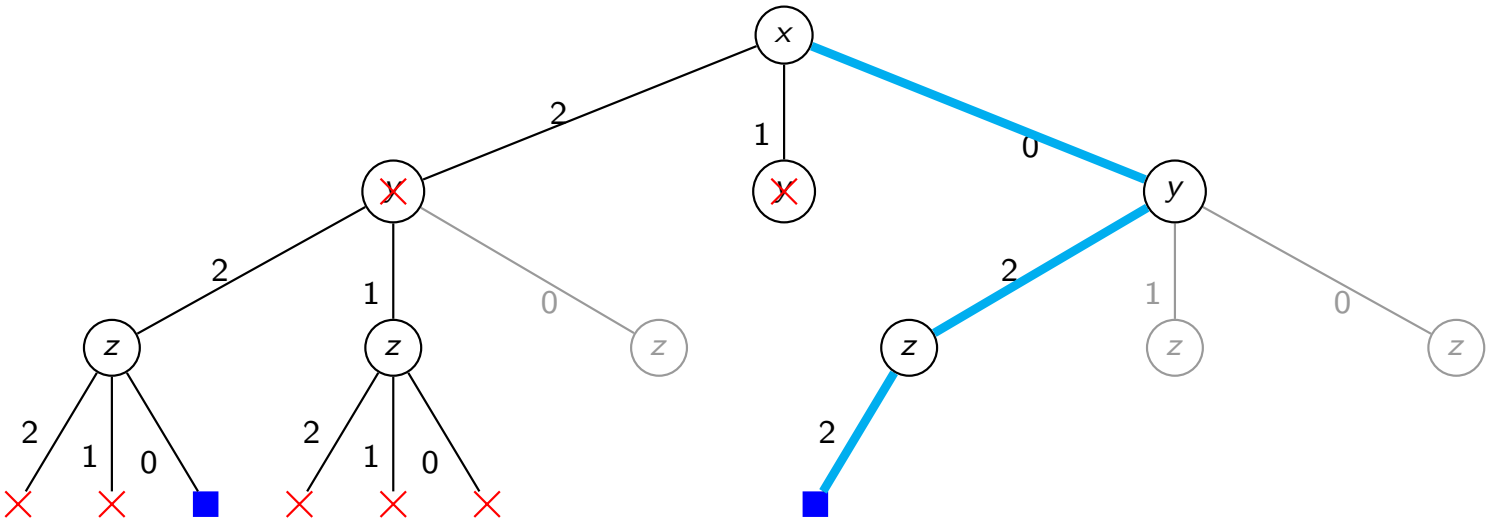
$$\exists x \forall y \exists z (x + z \leq y)$$



$$\boxed{0 + 2 \leq 2}$$

Résolution d'un QCSP

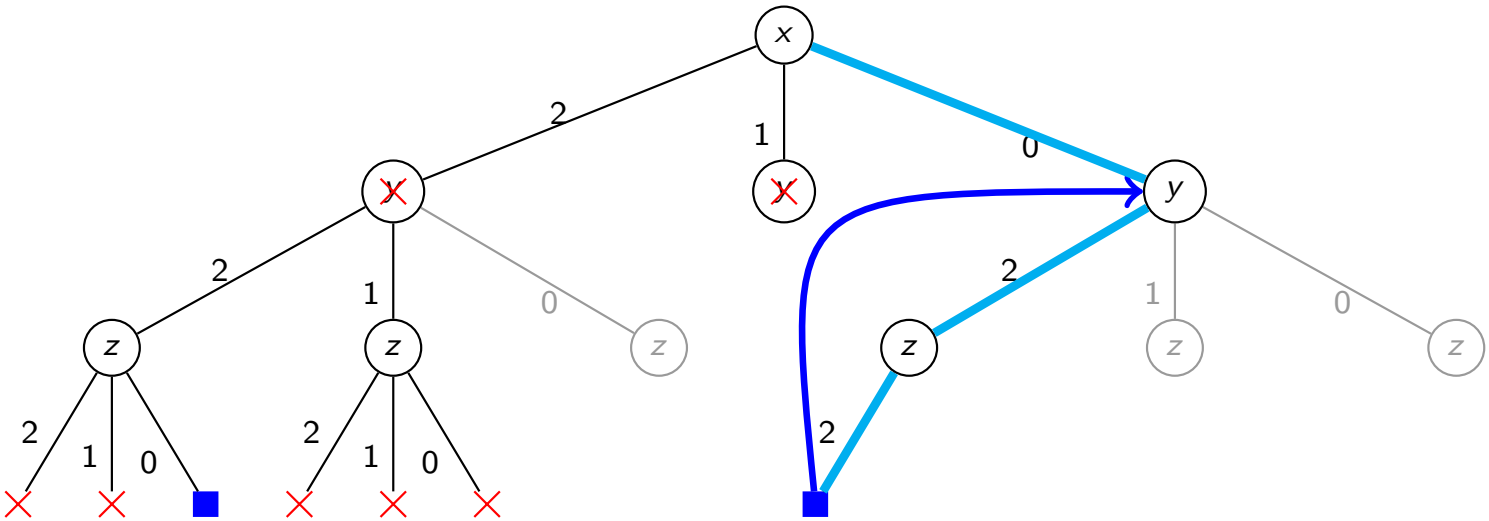
$$\exists x \forall y \exists z (x + z \leq y)$$



Scénario gagnant : $(x = 0)$, $(y = 2)$, $(z = 2)$

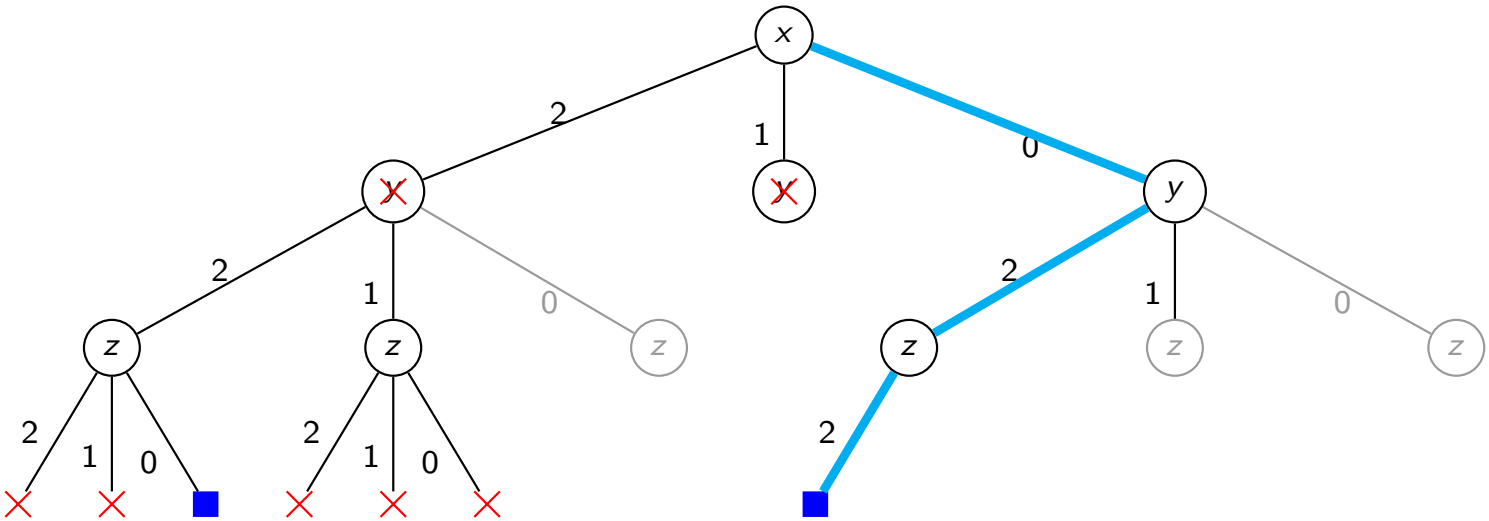
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



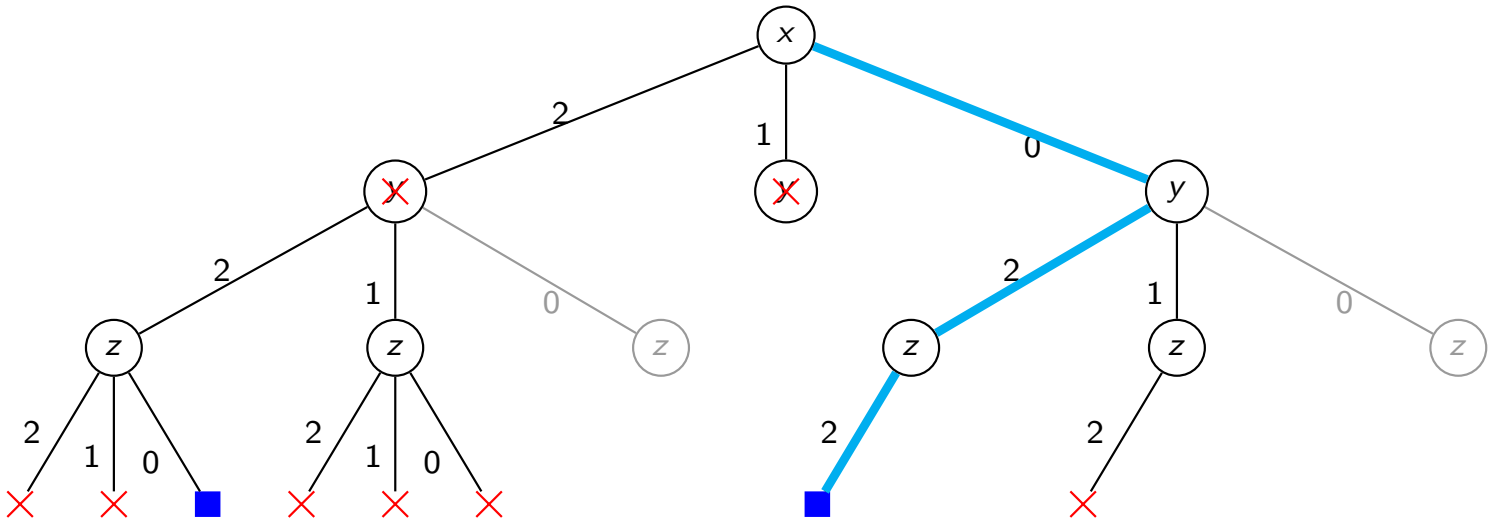
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



Résolution d'un QCSP

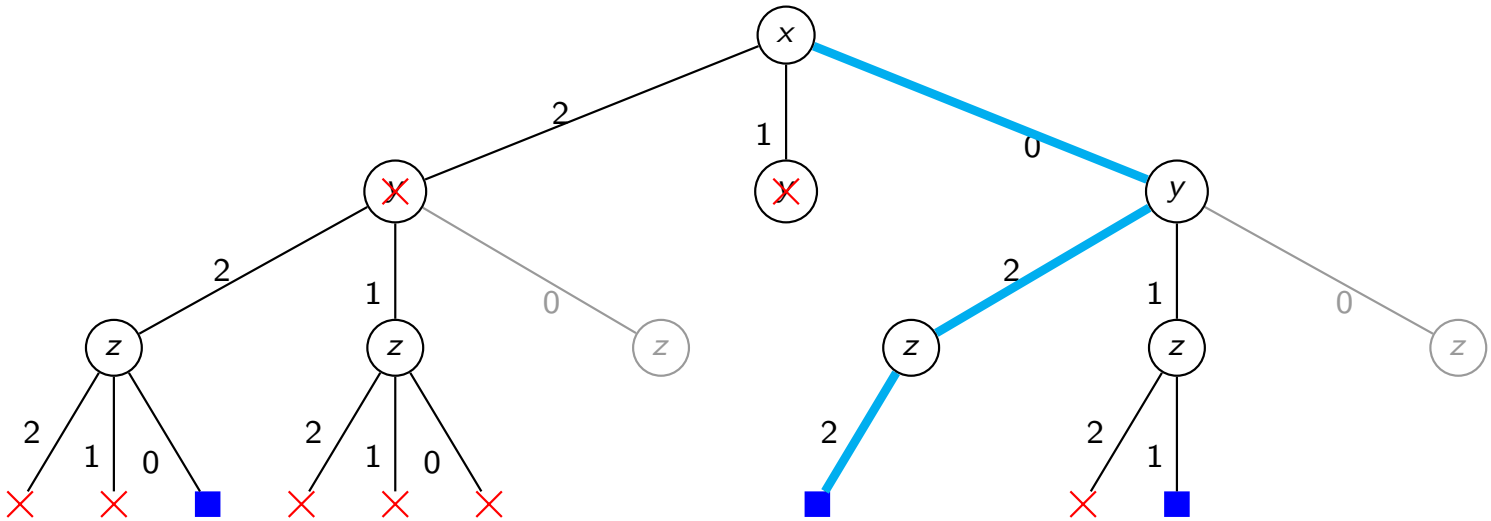
$$\exists x \forall y \exists z (x + z \leq y)$$



$0 + 2 > 1$

Résolution d'un QCSP

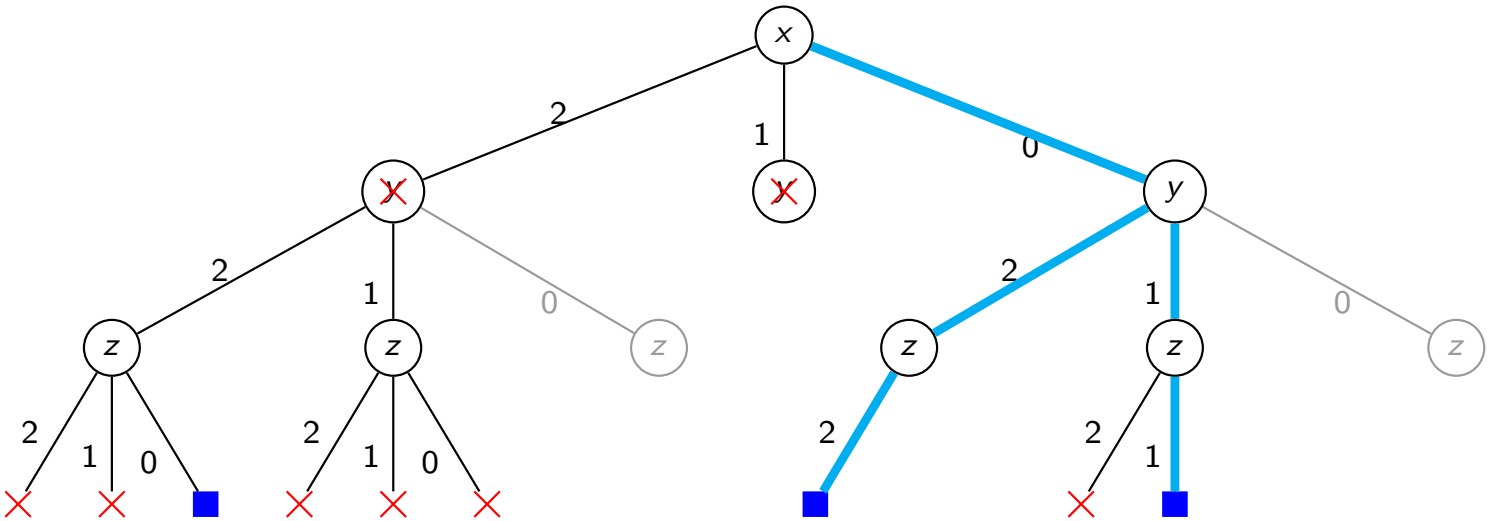
$$\exists x \forall y \exists z (x + z \leq y)$$



$$0 + 2 \leq 2$$

Résolution d'un QCSP

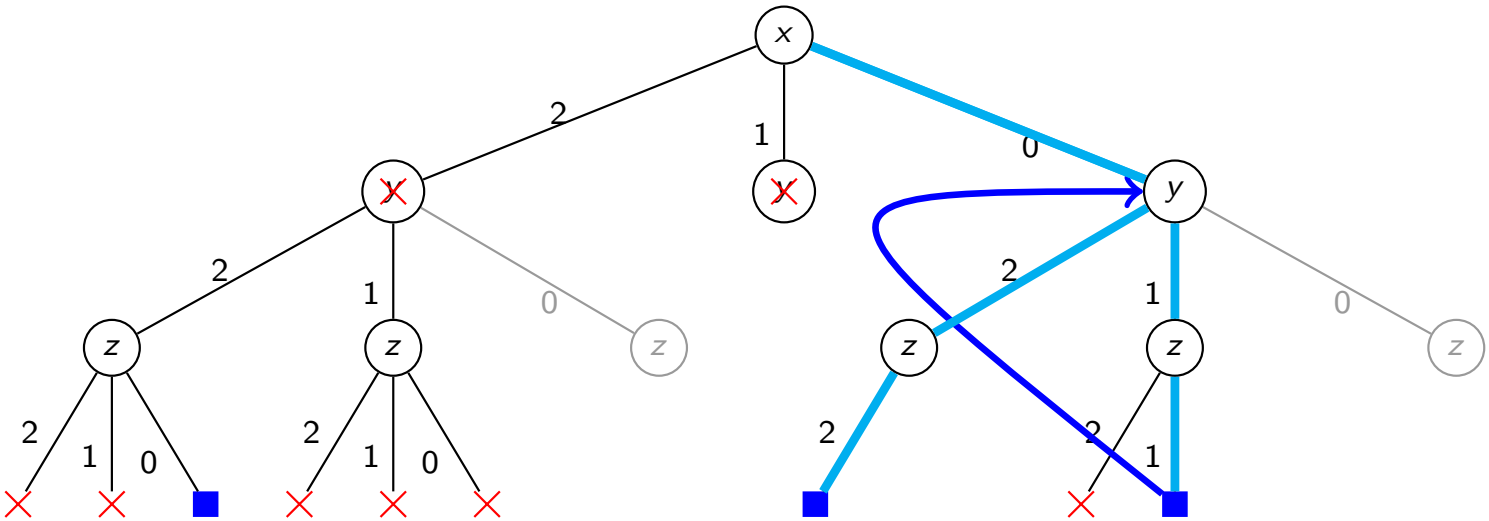
$$\exists x \forall y \exists z (x + z \leq y)$$



Scénario gagnant : $(x = 0)$, $(y = 1)$, $(z = 1)$

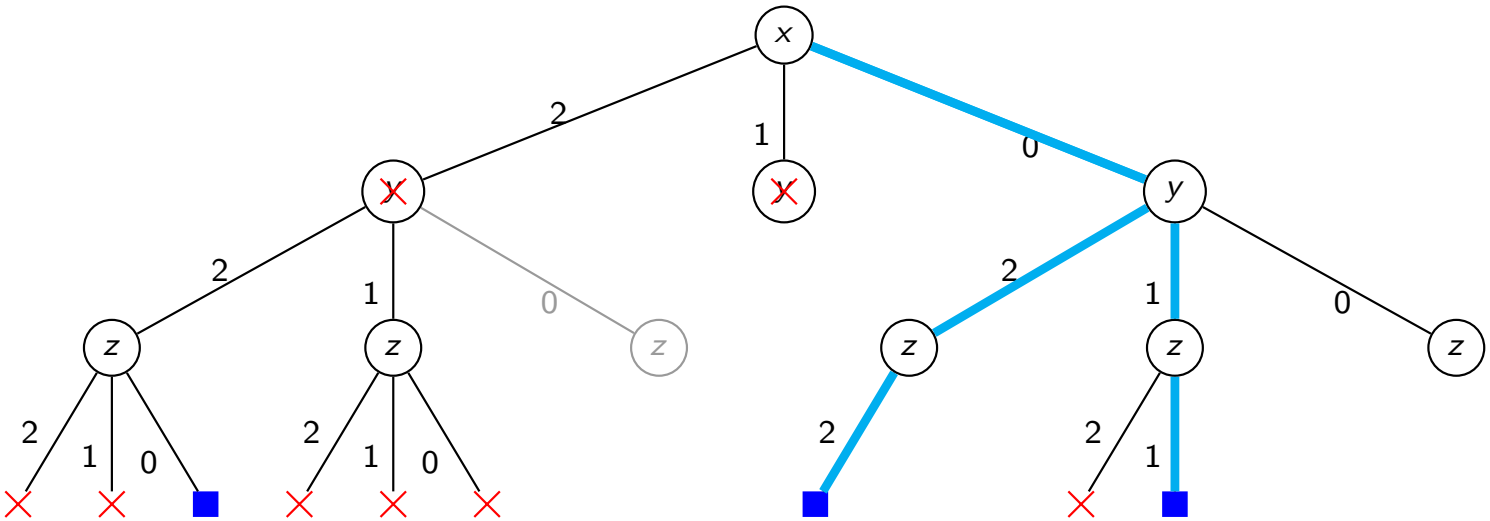
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



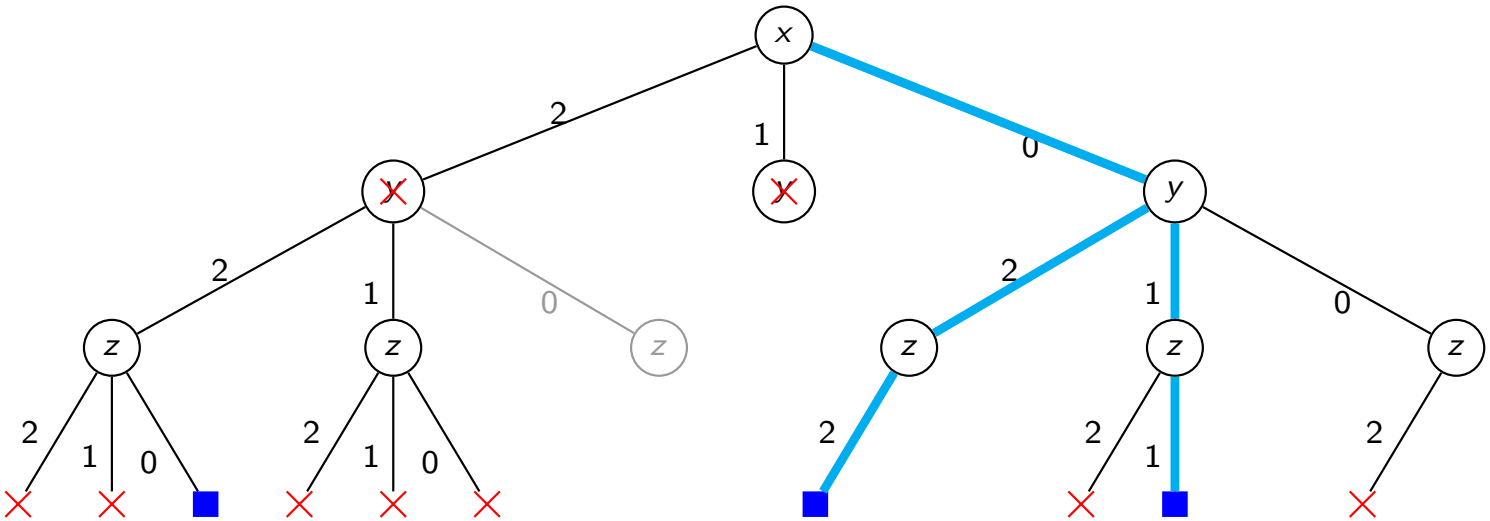
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



Résolution d'un QCSP

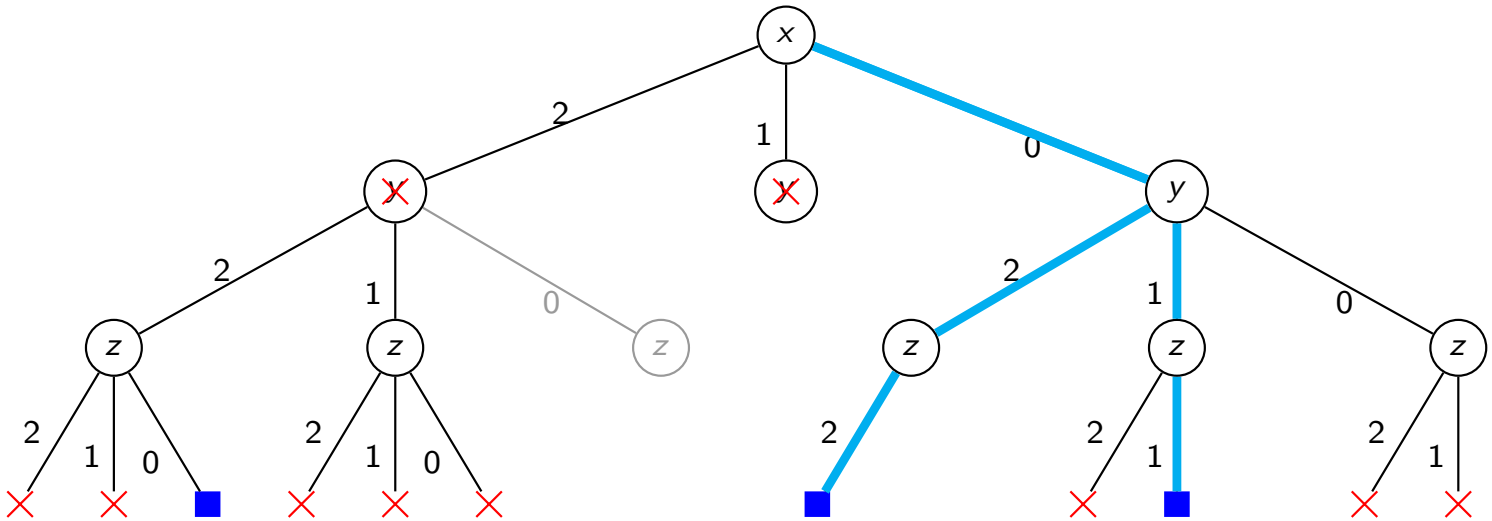
$$\exists x \forall y \exists z (x + z \leq y)$$



$0 + 2 > 0$

Résolution d'un QCSP

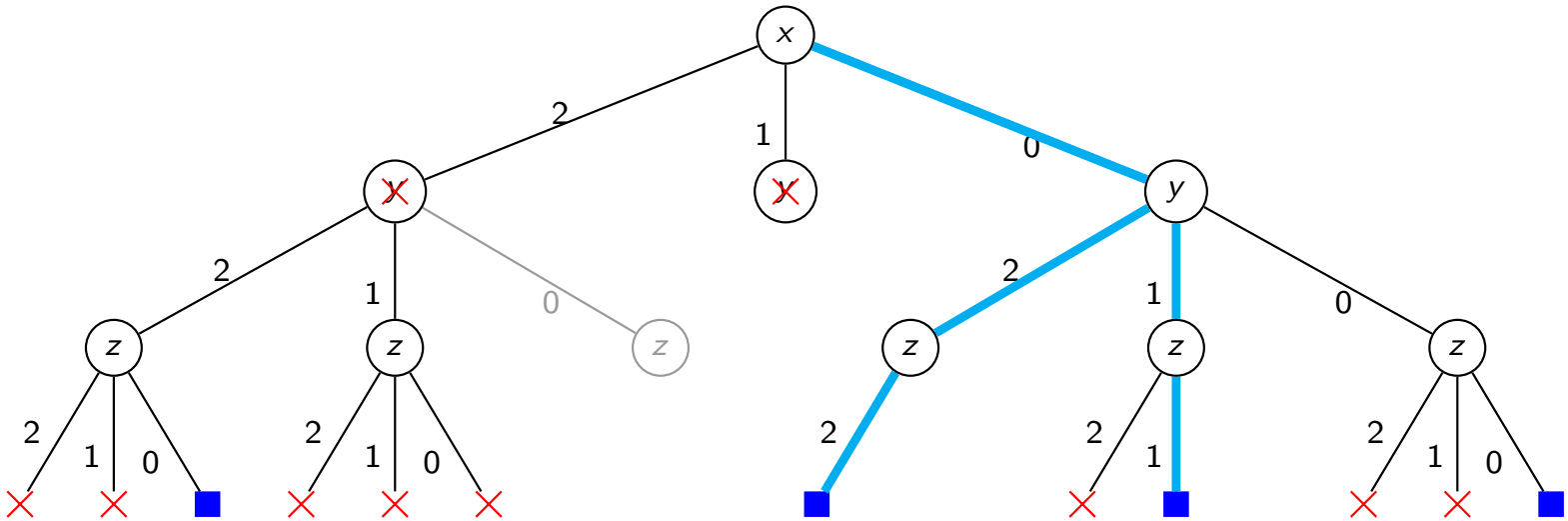
$$\exists x \forall y \exists z (x + z \leq y)$$



$0 + 1 > 0$

Résolution d'un QCSP

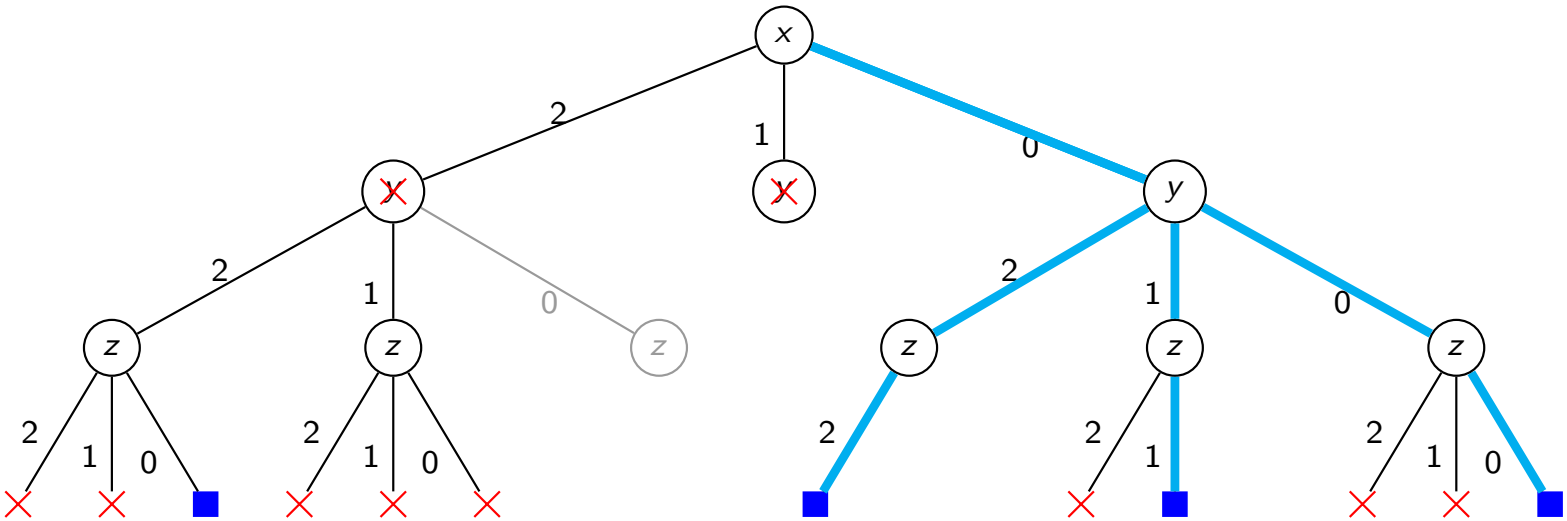
$$\exists x \forall y \exists z (x + z \leq y)$$



$$0 + 0 \leq 0$$

Résolution d'un QCSP

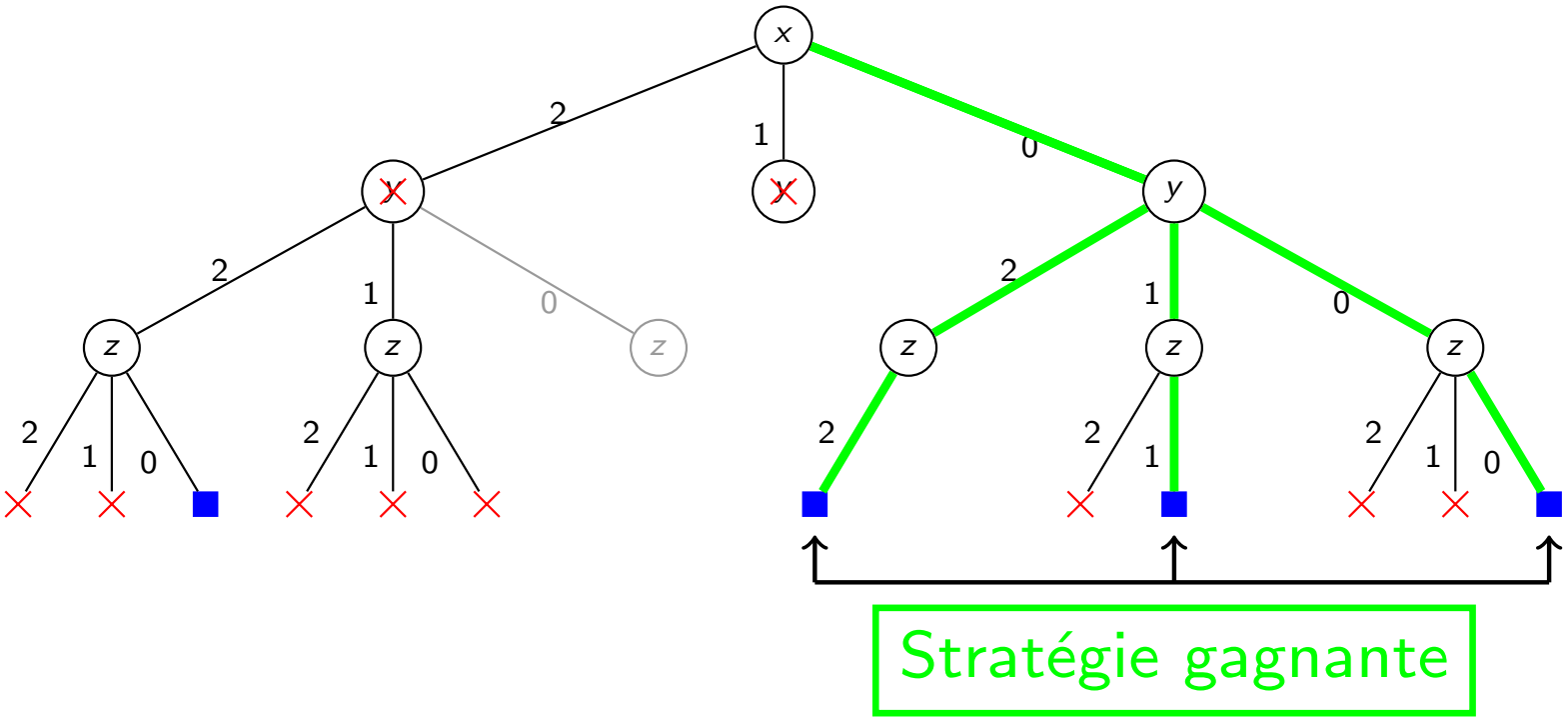
$$\exists x \forall y \exists z (x + z \leq y)$$



Scénario gagnant : $(x = 0)$, $(y = 0)$, $(z = 0)$

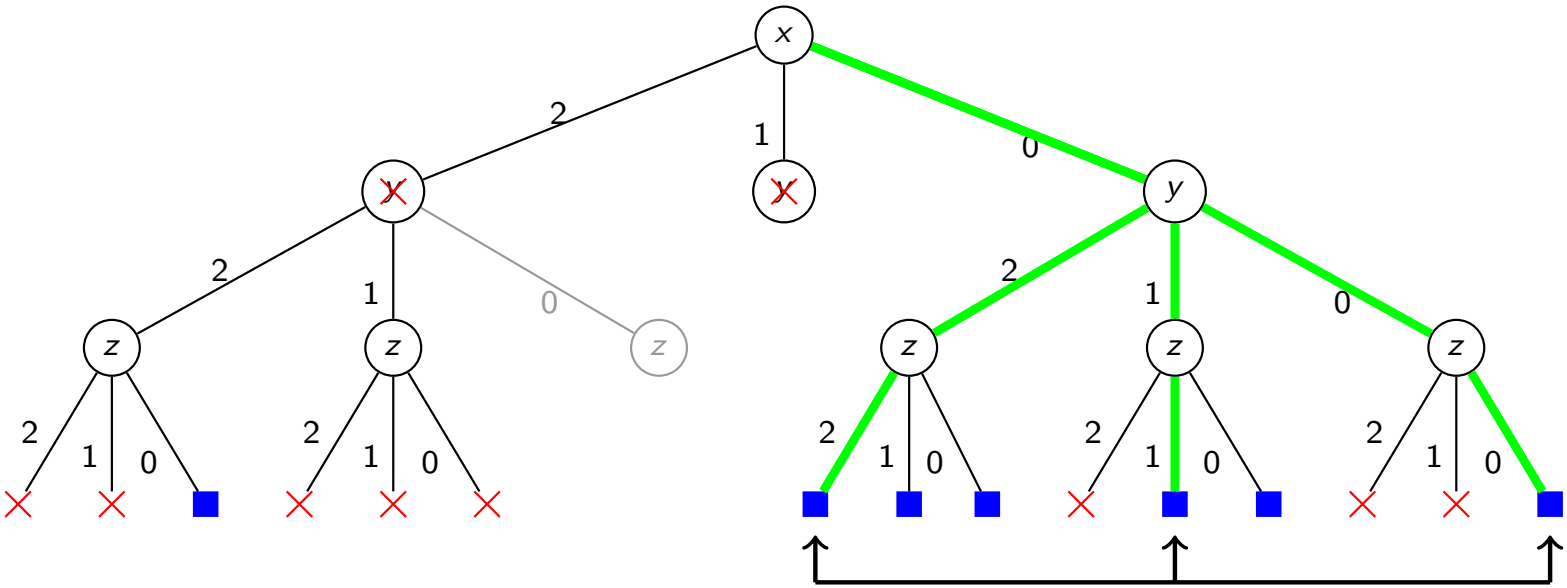
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



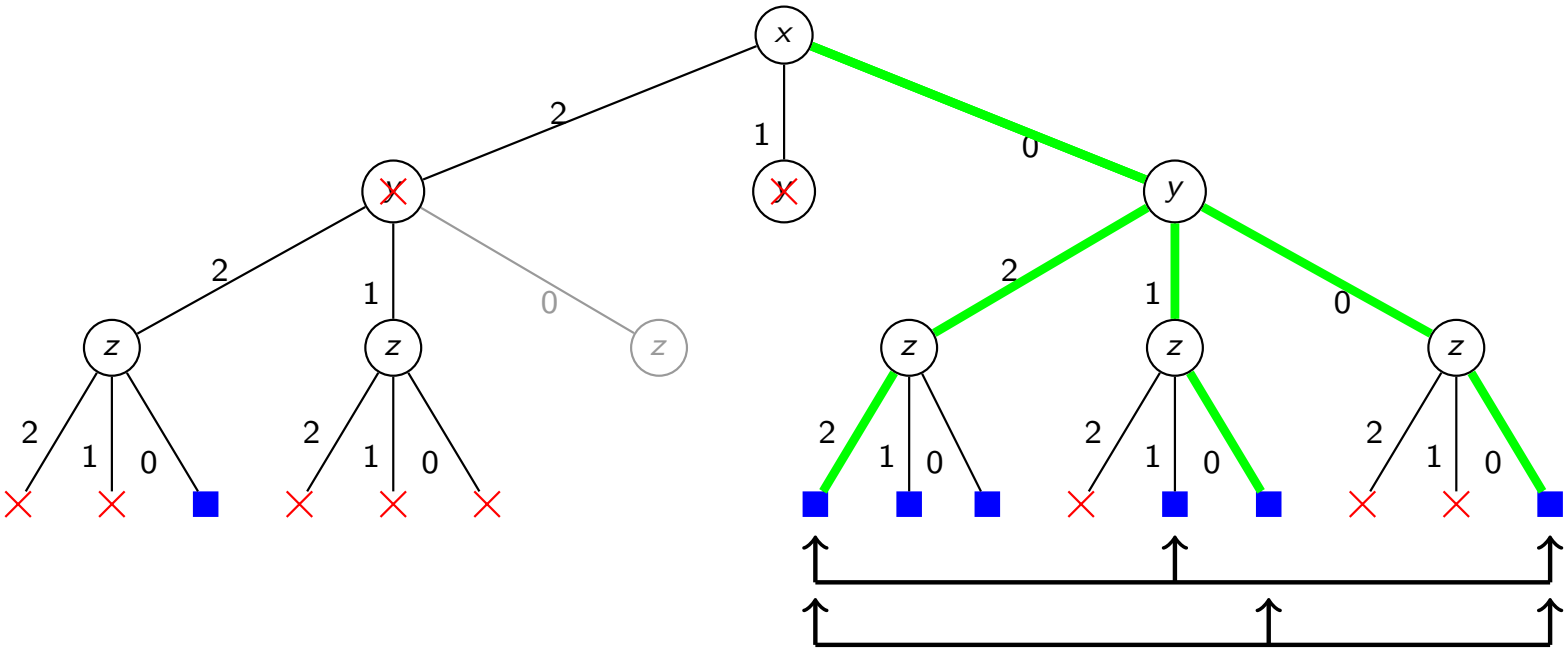
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



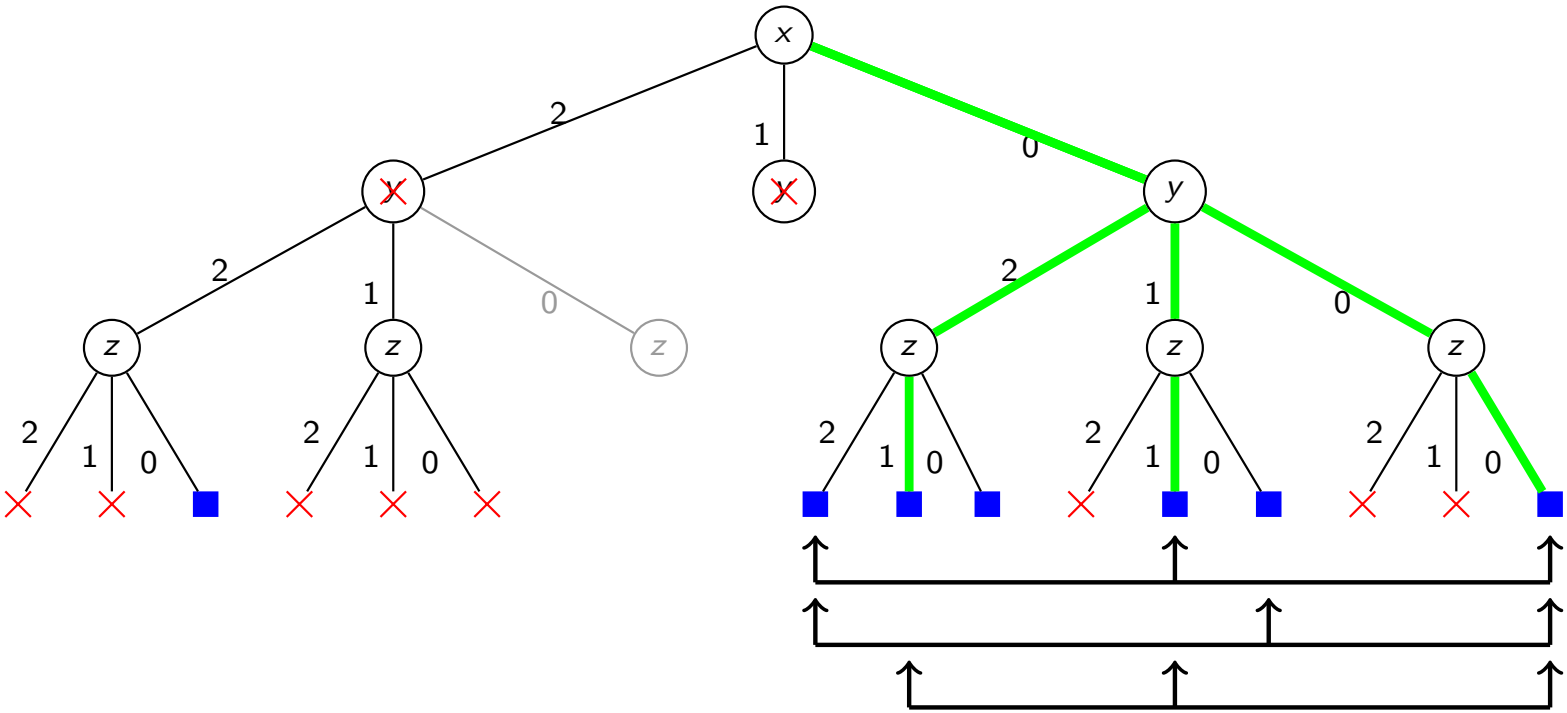
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



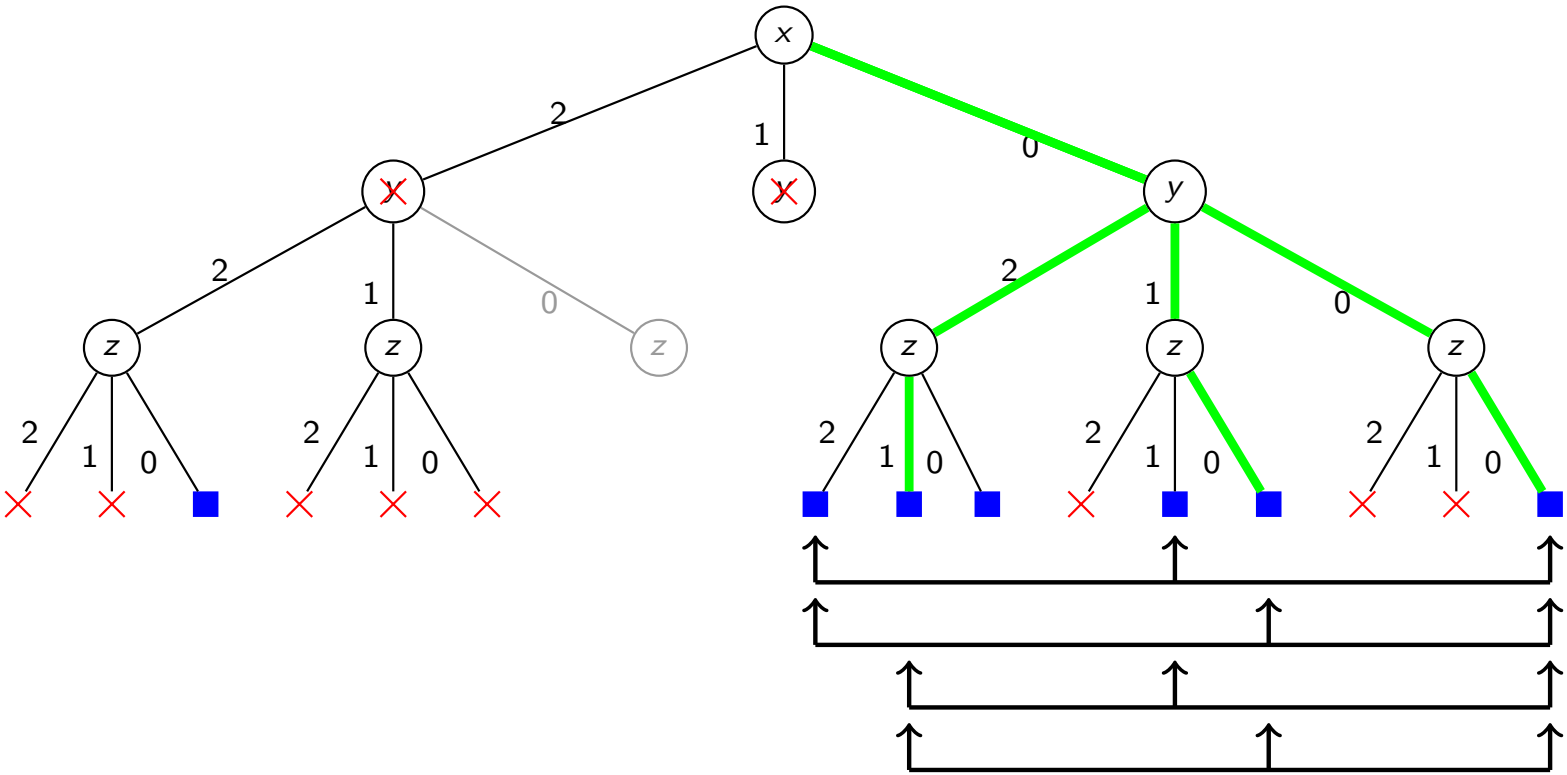
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



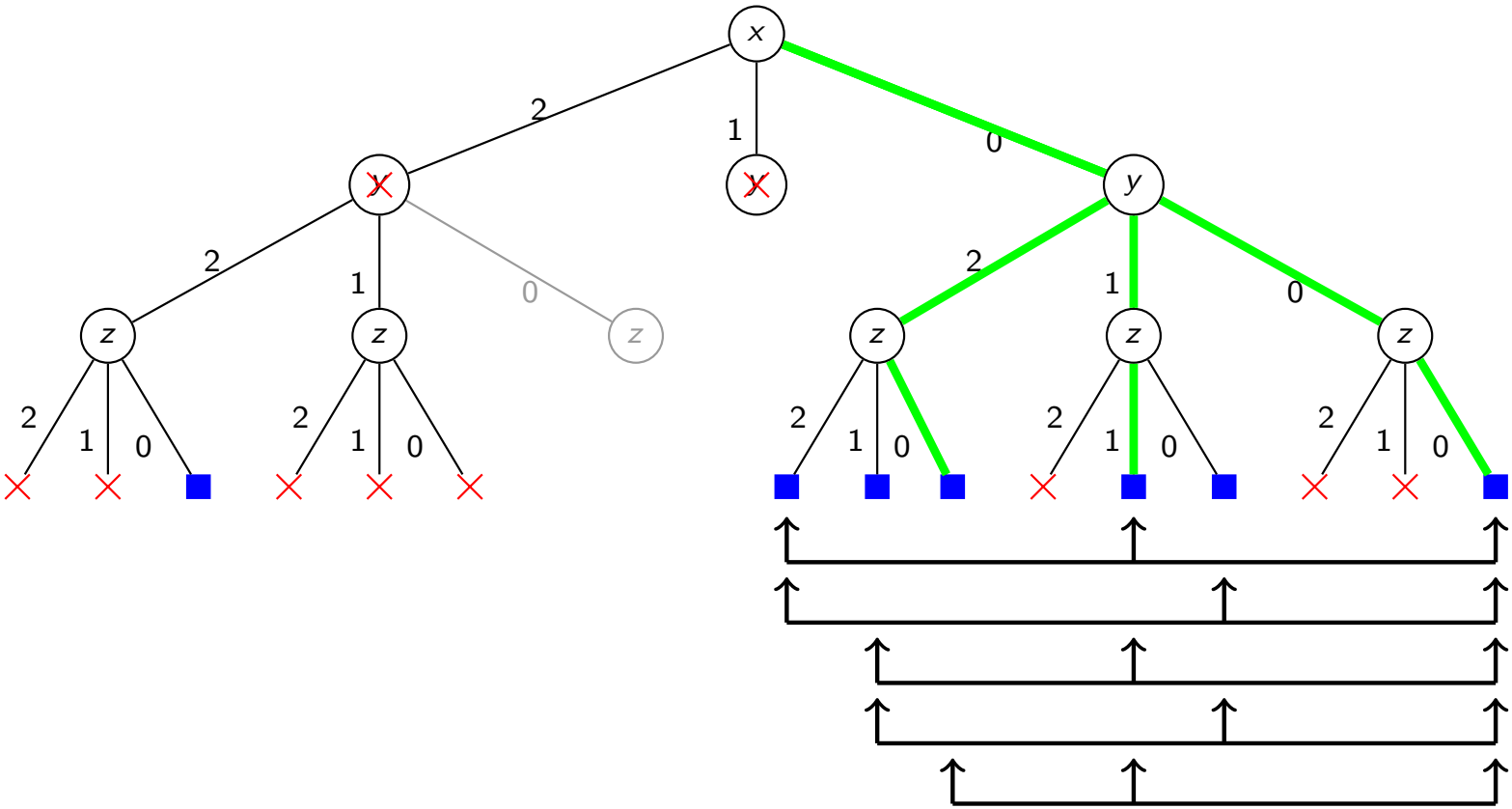
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



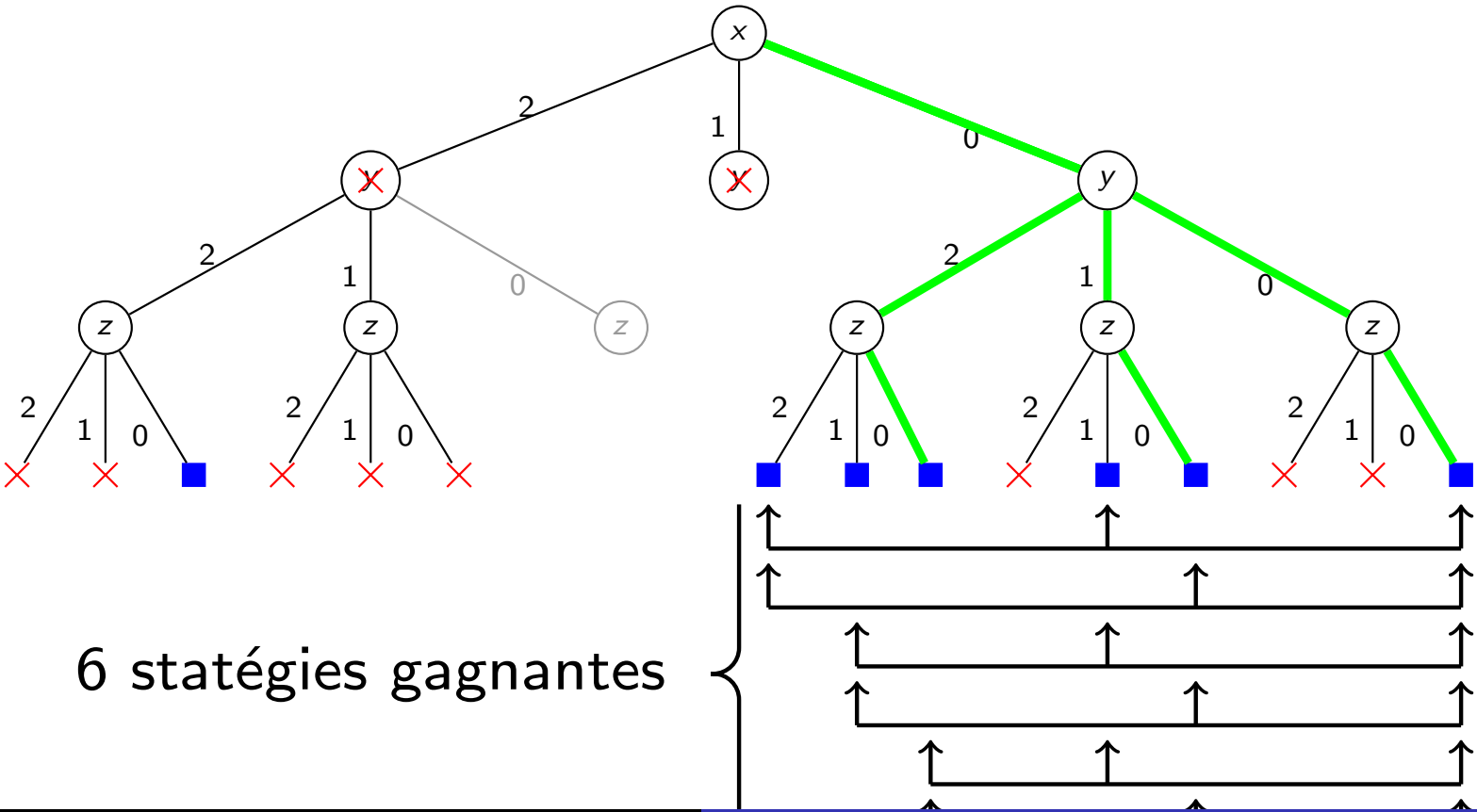
Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



Résolution d'un QCSP

$$\exists x \forall y \exists z (x + z \leq y)$$



6 statégies gagnantes

Plan

Complexité algorithmique

- La solution d'un QCSP (stratégie gagnante) est un arbre.

Complexité algorithmique

- La solution d'un QCSP (stratégie gagnante) est un arbre.
- Tout candidat solution exige un test de vérification appartenant à la classe de complexité co-NP-COMPLET.

Complexité algorithmique

- La solution d'un QCSP (stratégie gagnante) est un arbre.
- Tout candidat solution exige un test de vérification appartenant à la classe de complexité co-NP-COMPLET.
- Les QCSP appartiennent à la classe de complexité PSPACE

Problématique

- Il est communément admis que les problèmes de la classe PSPACE sont beaucoup plus difficiles à résoudre en temps que les problèmes de la classe NP.

Problématique

- Il est communément admis que les problèmes de la classe PSPACE sont beaucoup plus difficiles à résoudre en temps que les problèmes de la classe NP.
- Les instances QCSP de grande tailles ne peuvent pas être résolues en un temps raisonnable par un solveur seul.

Problématique

- Il est communément admis que les problèmes de la classe PSPACE sont beaucoup plus difficiles à résoudre en temps que les problèmes de la classe NP.
- Les instances QCSP de grande tailles ne peuvent pas être résolues en un temps raisonnable par un solveur seul.
- Les méta-heuristiques sont efficaces uniquement si leurs fonctions d'évaluation sont rapides.

Problématique

- Il est communément admis que les problèmes de la classe PSPACE sont beaucoup plus difficiles à résoudre en temps que les problèmes de la classe NP.
- Les instances QCSP de grande tailles ne peuvent pas être résolues en un temps raisonnable par un solveur seul.
- Les méta-heuristiques sont efficaces uniquement si leurs fonctions d'évaluation sont rapides.
- Or la vérification d'un candidat solution est co-NP-COMPLET.

Problématique

- Il est communément admis que les problèmes de la classe PSPACE sont beaucoup plus difficiles à résoudre en temps que les problèmes de la classe NP.
- Les instances QCSP de grande tailles ne peuvent pas être résolues en un temps raisonnable par un solveur seul.
- Les méta-heuristiques sont efficaces uniquement si leurs fonctions d'évaluation sont rapides.
- Or la vérification d'un candidat solution est co-NP-COMPLET.
- Les méta-heuristiques ne peuvent pas résoudre à elles seules des instances QCSP de grande tailles.

Propositions

- Nous proposons ici une coopération entre le solveur de QCSP QuaCode et une heuristique de type Monte Carlo.

Propositions

- Nous proposons ici une coopération entre le solveur de QCSP QuaCode et une heuristique de type Monte Carlo.
- Cette heuristique se focalise sur l'espace de recherche contenant les non solutions (scénarios perdants), cela afin de pallier au coût co-NP-COMPLET de la vérification d'un candidat solution.

Propositions

- Nous proposons ici une coopération entre le solveur de QCSP QuaCode et une heuristique de type Monte Carlo.
- Cette heuristique se focalise sur l'espace de recherche contenant les non solutions (scénarios perdants), cela afin de pallier au coût co-NP-COMPLET de la vérification d'un candidat solution.
- Par dualité, nous pensons pouvoir cibler les zones de l'espace de recherche contenant des stratégies gagnantes.

Propositions

- Nous proposons ici une coopération entre le solveur de QCSP QuaCode et une heuristique de type Monte Carlo.
- Cette heuristique se focalise sur l'espace de recherche contenant les non solutions (scénarios perdants), cela afin de pallier au coût co-NP-COMPLET de la vérification d'un candidat solution.
- Par dualité, nous pensons pouvoir cibler les zones de l'espace de recherche contenant des stratégies gagnantes.
- Ainsi, l'heuristique de type Monte Carlo pourra guider la résolution de QuaCode en réordonnant le parcours des domaines des variables du problème.

Heuristique de type Monte Carlo

- D'abord, l'algorithme instancie toutes les variables de façon aléatoire.

Heuristique de type Monte Carlo

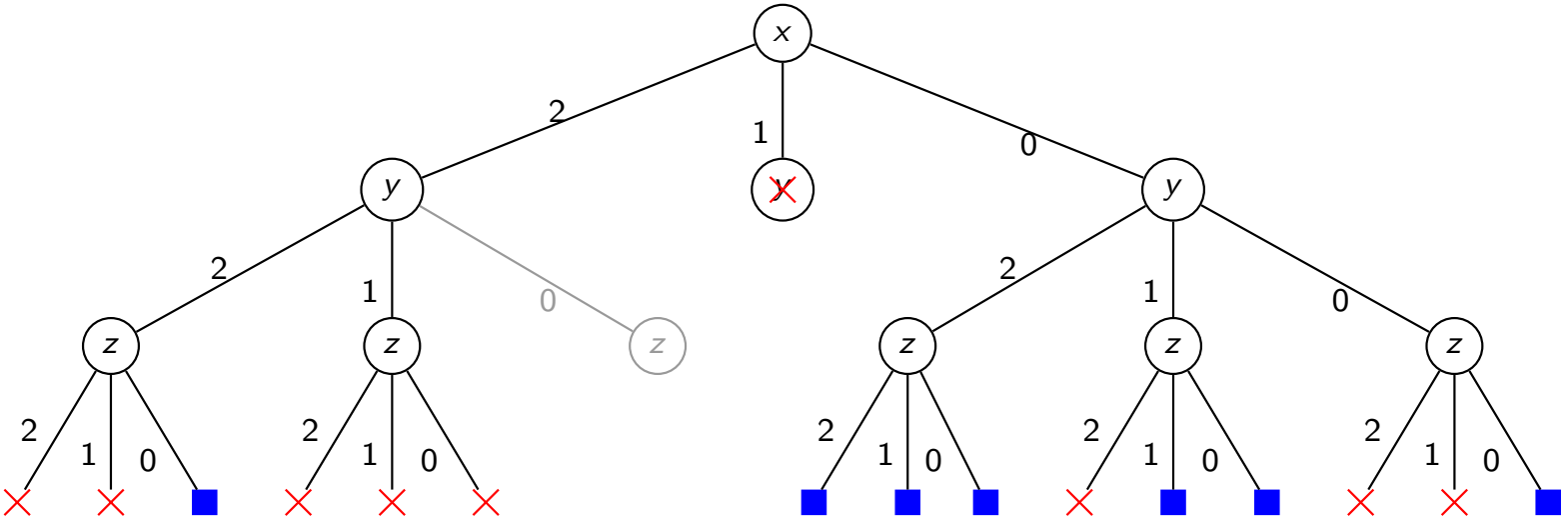
- D'abord, l'algorithme instancie toutes les variables de façon aléatoire.
- Ensuite, il entre dans une boucle perpétuelle qui va modifier l'instance courante de manière à la dégrader (recherche des non solutions).

Heuristique de type Monte Carlo

- D'abord, l'algorithme instancie toutes les variables de façon aléatoire.
- Ensuite, il entre dans une boucle perpétuelle qui va modifier l'instance courante de manière à la dégrader (recherche des non solutions).
- Dans le même temps, l'algorithme établit des statistiques sur le nombre de conflit que présentent les valeurs des variables sur les diverses instanciations.

Heuristique de type Monte Carlo

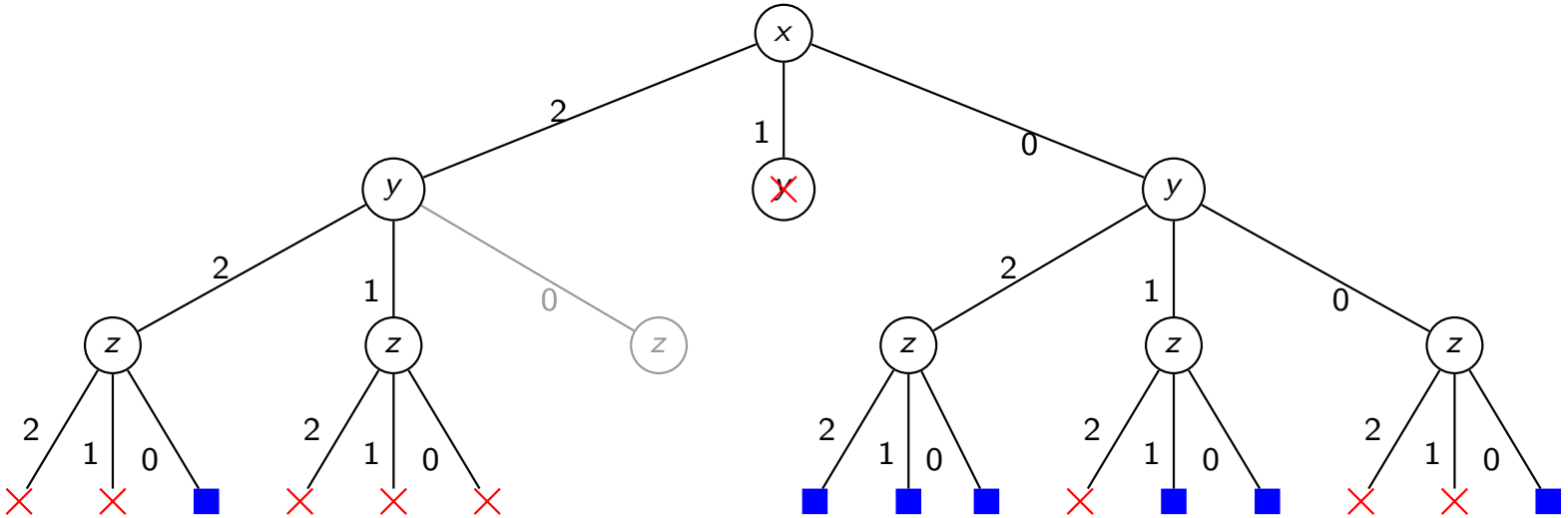
- D'abord, l'algorithme instancie toutes les variables de façon aléatoire.
- Ensuite, il entre dans une boucle perpétuelle qui va modifier l'instance courante de manière à la dégrader (recherche des non solutions).
- Dans le même temps, l'algorithme établit des statistiques sur le nombre de conflit que présentent les valeurs des variables sur les diverses instanciations.
- Après un certains nombre d'itérations dans cette boucle, l'algorithme établit un ordre sur les valeurs des variables :



Ordre sur les valeurs des variables

Les valeurs des domaines des variables sont triées par ordre :

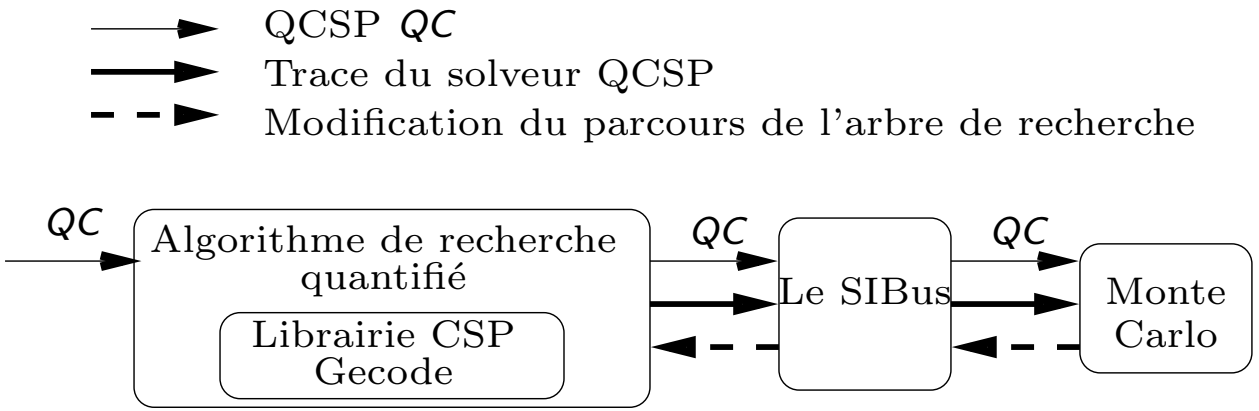
- Croissant de conflits pour les variables quantifiées existentiellement.



Ordre sur les valeurs des variables

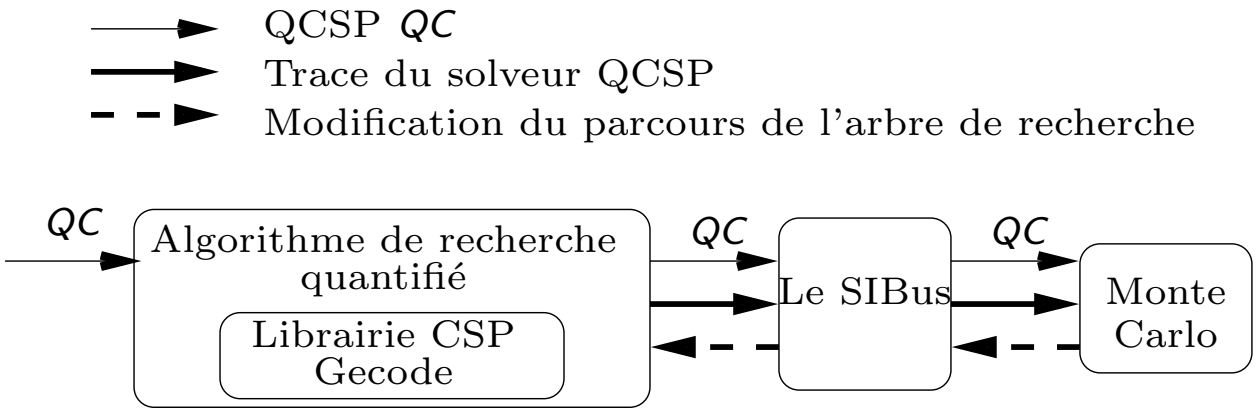
Les valeurs des domaines des variables sont triées par ordre :

- Croissant de conflits pour les variables quantifiées existentiellement.
- Décroissant de conflits pour les variables quantifiées universellement.



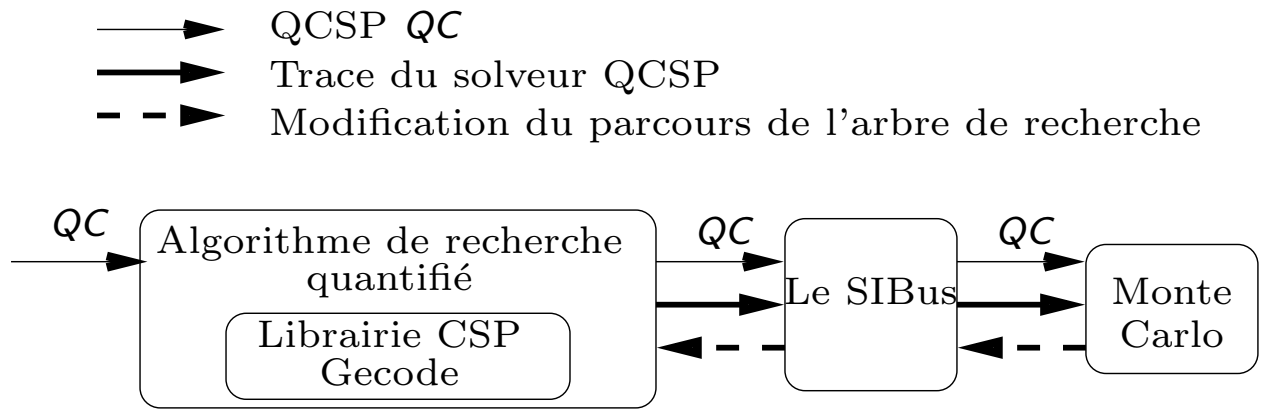
Mise en œuvre

- Notre heuristique établit des statistiques sur les conflits des valeurs des domaines de chacune des variables.



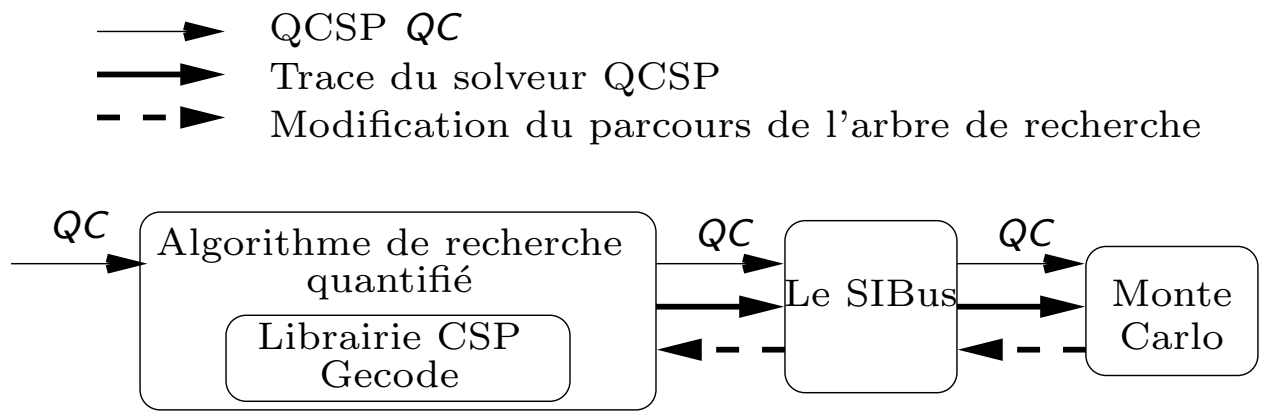
Mise en œuvre

- Notre heuristique établit des statistiques sur les conflits des valeurs des domaines de chacune des variables.
- Cet ordre de parcours des valeurs des domaines des variables est soumis fréquemment à QuaCode par le biais du SIBus.



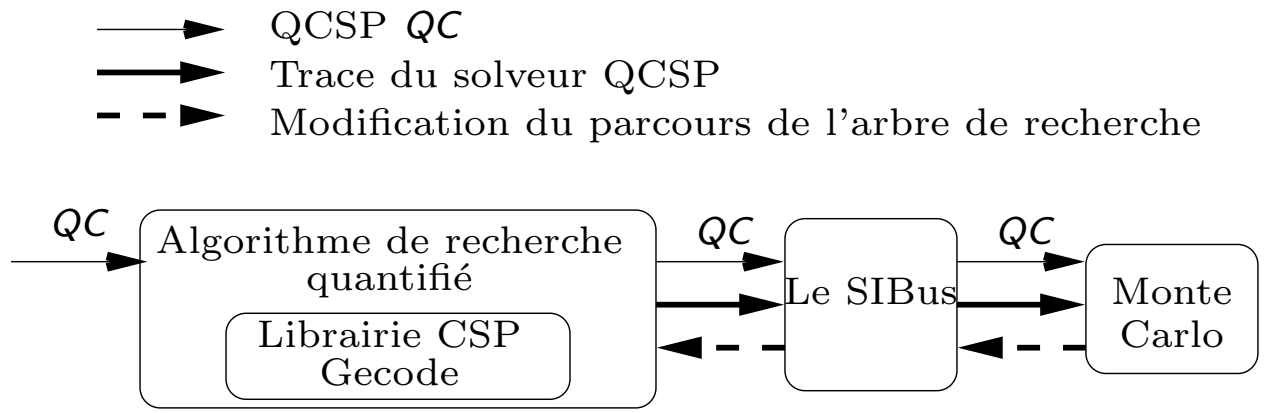
Mise en œuvre

- QuaCode modifie l'ordre de parcours des valeurs des domaines des variables en fonction de l'ordre établit par l'heuristique.



Mise en œuvre

- QuaCode modifie l'ordre de parcours des valeurs des domaines des variables en fonction de l'ordre établit par l'heuristique.
- Les branches étant susceptibles de contenir des stratégies gagnantes sont ainsi examinées en priorité.



Mise en œuvre

- QuaCode modifie l'ordre de parcours des valeurs des domaines des variables en fonction de l'ordre établit par l'heuristique.
- Les branches étant susceptibles de contenir des stratégies gagnantes sont ainsi examinées en priorité.
- La complétude est conservée puisque aucune valeur n'est omise lors du nouveau parcours.

Expérimentations

Problème du boulanger

Un boulanger souhaite acquérir quatre poids à choisir dans l'intervalle $\{1, \ldots, 40\}$ kg, lui permettant de peser **n'importe quelle** quantité entière de farine dans l'intervalle $\{1, \ldots, 40\}$ kg en utilisant une balance de Roberval. Pour la pesée, chaque poids peut être placé de n'importe quel côté de la balance ou ne pas être utilisé.

Formalisation du problème

Soit $w_1, w_2, w_3, w_4, f \in \{1, \ldots, 40\}$,
et $c_1, c_2, c_3, c_4 \in \{-1, 0, 1\}$,

$$\exists w_1 \exists w_2 \exists w_3 \exists w_4 \forall f \exists c_1 \exists c_2 \exists c_3 \exists c_4 (\sum_{i=1}^{4} w_i . c_i = f)$$

Formalisation du problème

Soit $w_1, w_2, w_3, w_4, f \in \{1, \ldots, 40\}$,
et $c_1, c_2, c_3, c_4 \in \{-1, 0, 1\}$,

$$\exists w_1 \exists w_2 \exists w_3 \exists w_4 \forall f \exists c_1 \exists c_2 \exists c_3 \exists c_4 (\sum_{i=1}^{4} w_i.c_i = f)$$

- Les w_i représentent les poids à acquérir, f est le poids du sac de farine et les c_i correspondent à l'endroit où l'on place les poids sur la balance de roberval.

Formalisation du problème

Soit $w_1, w_2, w_3, w_4, f \in \{1, \ldots, 40\}$,
et $c_1, c_2, c_3, c_4 \in \{-1, 0, 1\}$,

$$\exists w_1 \exists w_2 \exists w_3 \exists w_4 \forall f \exists c_1 \exists c_2 \exists c_3 \exists c_4 (\sum_{i=1}^{4} w_i.c_i = f)$$

- Les w_i représentent les poids à acquérir, f est le poids du sac de farine et les c_i correspondent à l'endroit où l'on place les poids sur la balance de roberval.
- Le problème du boulanger admet une unique stratégie gagnante $\{1, 3, 9, 27\}$

Formalisation du problème

Soit $w_1, w_2, w_3, w_4, f \in \{1, \ldots, 40\}$,
et $c_1, c_2, c_3, c_4 \in \{-1, 0, 1\}$,

$$\exists w_1 \exists w_2 \exists w_3 \exists w_4 \forall f \exists c_1 \exists c_2 \exists c_3 \exists c_4 (\sum_{i=1}^{4} w_i.c_i = f)$$

- Les w_i représentent les poids à acquérir, f est le poids du sac de farine et les c_i correspondent à l'endroit où l'on place les poids sur la balance de roberval.
- Le problème du boulanger admet une unique stratégie gagnante $\{1, 3, 9, 27\}$
- Nous créons une nouvelle instance en restreignant le domaine de w_1 à $\{2, \ldots, 40\}$.

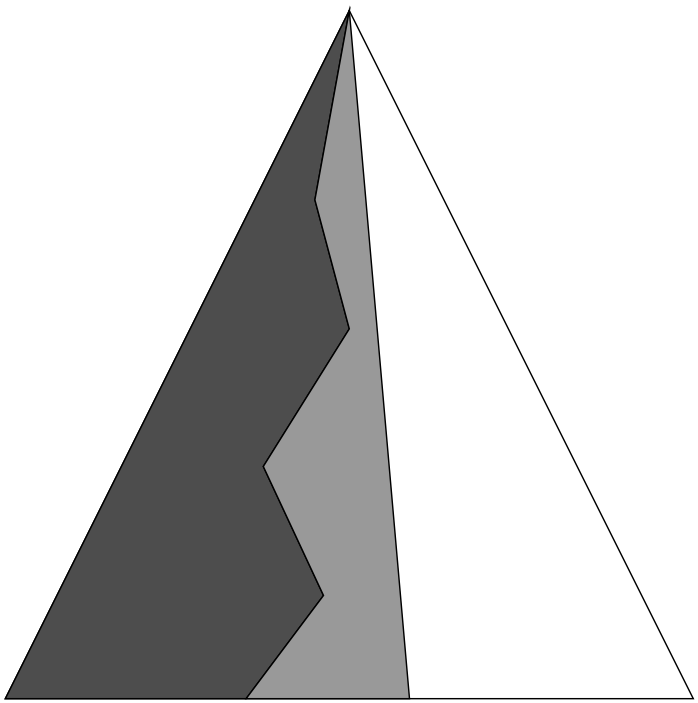
Résultats

		Temps (ms)			Nœuds explorés		
		Moy.	Min	Max	Moy.	Min	Max
Instance 1	QuaCode	3971	3822	4137	19130	19130	19130
	QuaCode+MC	1457	1021	2593	13453	7661	27988
Instance 2	QuaCode	152341	146377	156286	1171084	1171084	1171084
	QuaCode+MC	70151	1498	132221	901853	14662	1697205

1. Prise en compte de la trace du solveur QuaCode

Espace déjà parcouru par l'algorithme complet

Frontière

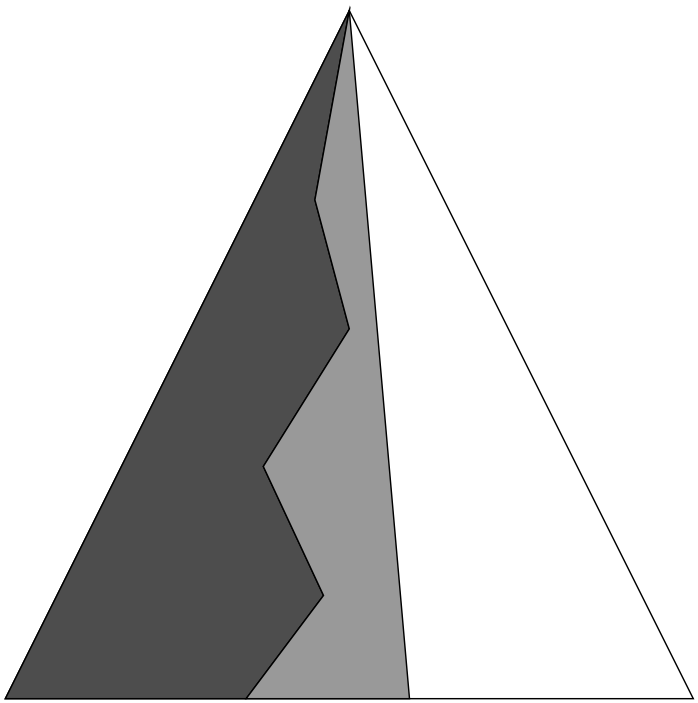


- Empêcher l'heuristique de type Monte Carlo de tester des zones obsolètes de l'espace de recherche

1. Prise en compte de la trace du solveur QuaCode

Espace déjà parcouru par l'algorithme complet

Frontière



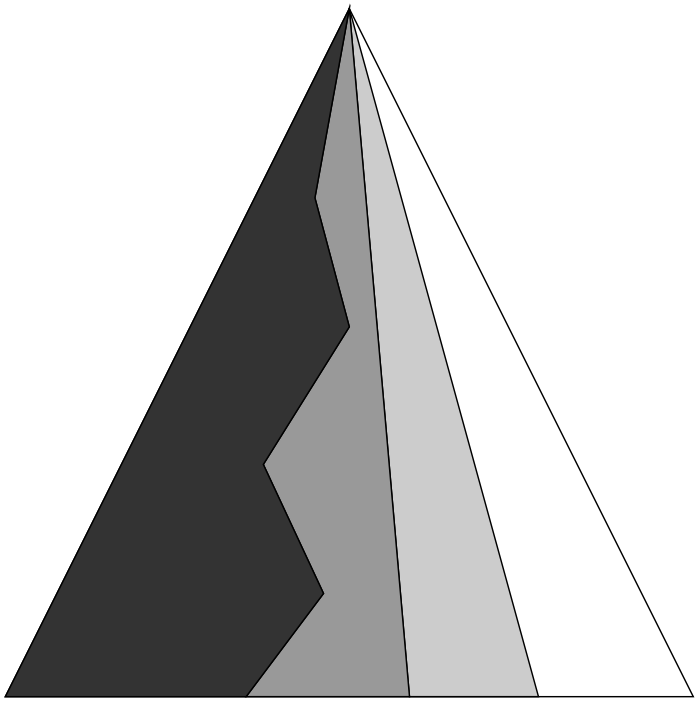
- Empêcher l'heuristique de type Monte Carlo de tester des zones obsolètes de l'espace de recherche
- Obtenir des statistiques plus pertinentes

2. Intensifier la recherche au voisinage de la frontière

Espace déjà parcouru par l'algorithme complet

Frontière

Espace de recherche de l'algorithme de type Monte Carlo



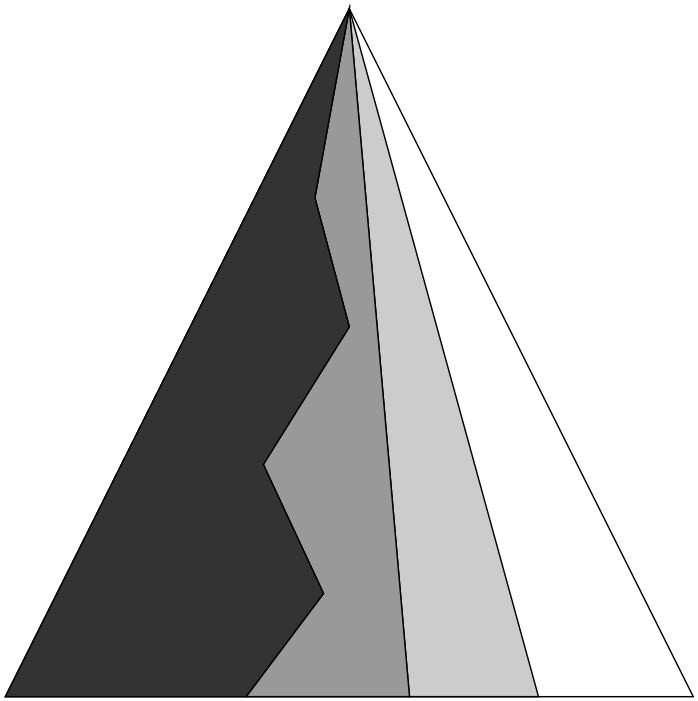
- Éviter de prospecter une zone qui ne sera probablement pas examinée par QuaCode

2. Intensifier la recherche au voisinage de la frontière

Espace déjà parcouru par l'algorithme complet

Frontière

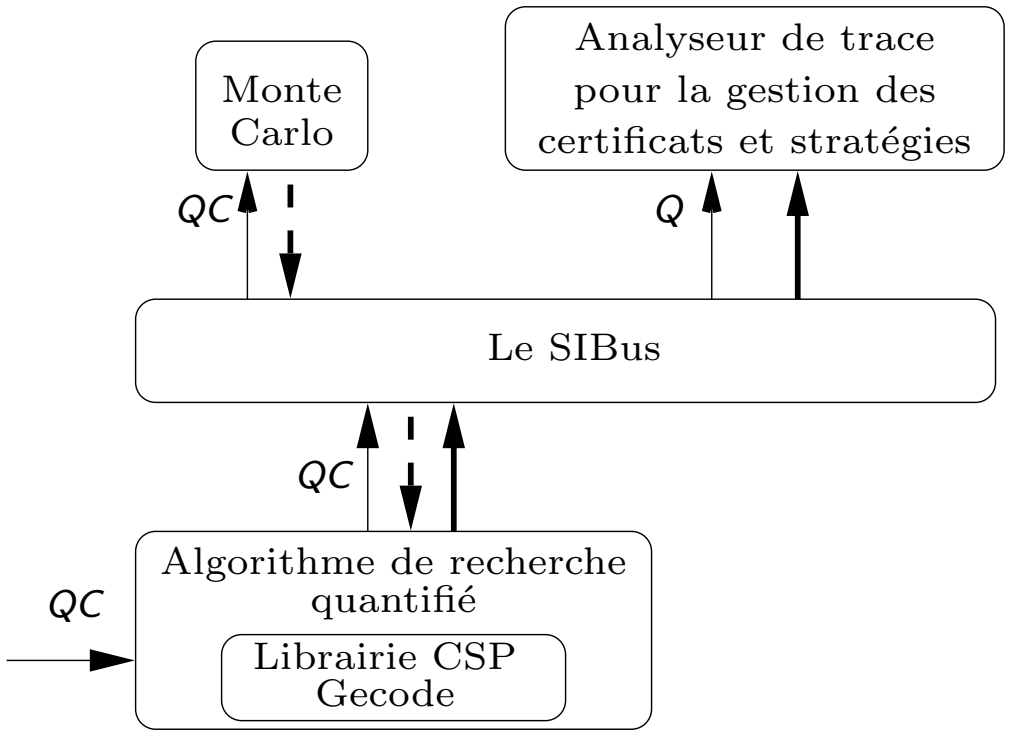
Espace de recherche de l'algorithme de type Monte Carlo



- Éviter de prospecter une zone qui ne sera probablement pas examinée par QuaCode
- Réduire l'espace de recherche de l'heuristique de type Monte Carlo

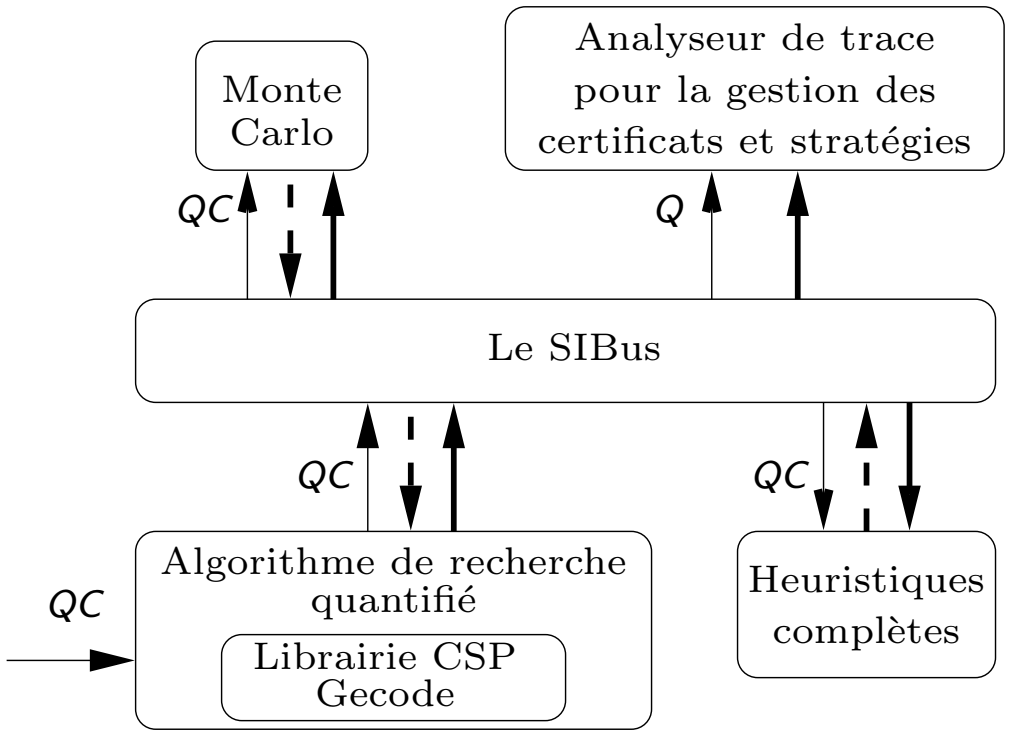
Architecture actuelle

QCSP *QC*
Trace du solveur QCSP
Modification du parcours de l'arbre de recherche



Architecture à venir

QCSP *QC*
Trace du solveur QCSP
Modification du parcours de l'arbre de recherche



Merci pour votre attention.