Synthesizing Filtering Algorithms for Global Chance-Constraints

Brahim Hnich[1]
Roberto Rossi[2]
S. Armagan Tarim[3]
Steven D. Prestwich[4]

[1]Faculty of Computer Science, Izmir University of Economics, Izmir, Turkey
[2]LDI, Wageningen UR, the Netherlands
[3]Dept. of Operations Management, Nottingham University Business School, UK
[4]Cork Constraint Computation Centre, University College Cork, Ireland

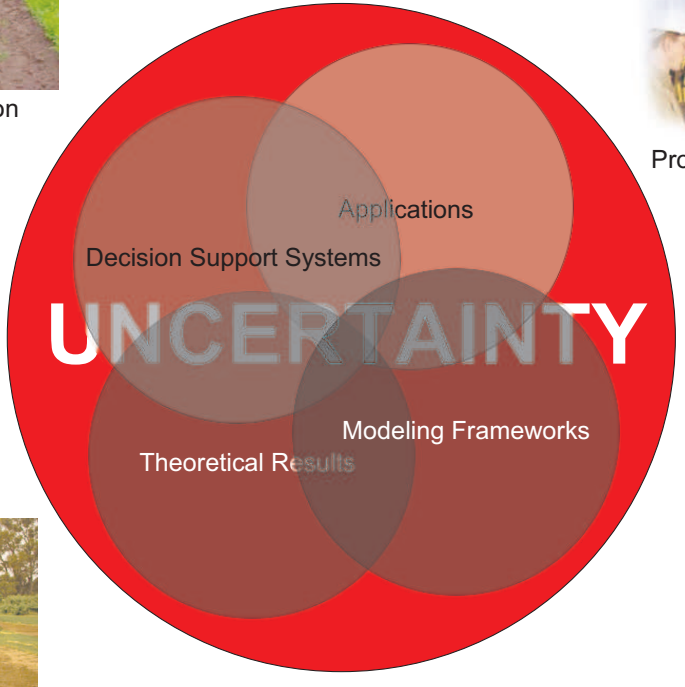
the 15th International Conference on Principles and Practice of Constraint Programming, CP-09

Decision Making Under Uncertainty: A Pervasive Issue

Land-Crop Allocation

Sustainable Energy Production

Food Quality Control



Production Planning

Financial Planning

Inventory Control

Decision Making Under Uncertainty: An Example

Static Stochastic Knapsack Problem

Problem: we have k kinds of **items** and a **knapsack** of size c into which to fit them. Each kind of item i has

- a **deterministic profit** r_i.
- a **size** w_i, which is not known at the time the decision has to be made. The decision maker knows the **probability distribution** of w_i.

A per unit **penalty cost** p has to be paid for exceeding the capacity of the knapsack. The **probability of not exceeding** the capacity of the knapsack should be greater or equal to a given **threshold** θ.
Objective: *find the knapsack that maximizes the expected profit.*

A slightly formal definition

Stochastic Constraint Satisfaction Problem (Walsh, 2002)

A Stochastic Constraint Satisfaction Problem (SCSP) is a 7-tuple

$$\langle V, S, D, P, C, \theta, L \rangle.$$

- $V = \{v_1, \ldots, v_n\}$ is a set of decision variables
- $S = \{s_1, \ldots, s_n\}$ is a set of stochastic variables
- D is a function mapping each variable to a domain of potential values
- P is a function mapping each variable in S to a probability distribution for its associated domain
- C is a set of (chance)-constraints, possibly involving stochastic variables
- θ_h is a threshold probability associated to chance-constraint h
- $L = [\langle V_1, S_1 \rangle, \ldots, \langle V_i, S_i \rangle, \ldots, \langle V_m, S_m \rangle]$ is a list of *decision stages*.

By considering an objective function $f(\widehat{V}, \widehat{S})$ we obtain a SCOP.

An Example

Sample SCOP: SSKP

- $V = \{x_1, \ldots, x_3\}$
- $D(x_i) = \{0, 1\} \quad \forall i \in \{1, \ldots, 3\}$
- $S = \{w_1, \ldots, w_3\}$
- $D(w_1) = \{5(0.5), 8(0.5)\}$,
  $D(w_2) = \{3(0.5), 9(0.5)\}$,
  $D(w_3) = \{15(0.5), 4(0.5)\}$
- $C = \{Pr(w_1x_1 + w_2x_2 + w_3x_3 \leq 10) \geq 0.2\}$
- $L = [\langle V, S \rangle]$
- $f(x_1, \ldots, x_3) =$
  $8x_1 + 15x_2 + 10x_3 - 2\mathbb{E}\max\left[0, \sum_{i=1}^{3} w_i x_i - 20\right]$

An Example

Sample SCOP: DSKP

- $V = \{x_1, \ldots, x_3\}$
- $D(x_i) = \{0, 1\} \quad \forall i \in \{1, \ldots, 3\}$
- $S = \{w_1, \ldots, w_3\}$
- $D(w_1) = \{5(0.5), 8(0.5)\}$,
  $D(w_2) = \{3(0.5), 9(0.5)\}$,
  $D(w_3) = \{15(0.5), 4(0.5)\}$
- $C = \{Pr(w_1x_1 + w_2x_2 + w_3x_3 \leq 10) \geq 0.2\}$
- $L = [\langle\{x_1\}, \{w_1\}\rangle, \langle\{x_2\}, \{w_2\}\rangle, \langle\{x_3\}, \{w_3\}\rangle]$
- $f(x_1, \ldots, x_3) =$
  $\mathbb{E}[8x_1 + 15x_2 + 10x_3] - 2\mathbb{E}\max\left[0, \sum_{i=1}^{3} w_i x_i - 20\right]$

Stochastic OPL

A **language** specifically introduced by Tarim et al. (Tarim et al., 2006) for **modeling decision problems under uncertainty**. It captures several **high level concepts** that facilitate the process of modeling uncertainty:

- stochastic variables (independent or conditional distributions)
- several probabilistic measures for the objective function (expectation, variance, etc.)
- chance-constraints
- decision stages
- ...

Stochastic OPL

SSKP

```
int N = 3;
int c = 10;
int p = 2;
float θ = 0.2
range Object [1..3];
int value[Object] = [8,15,10];
stoch int weight[Object] = [<5(0.5),8(0.5)>,
                    <3(0.5),9(0.5)>,<15(0.5),4(0.5)>];
var int+ X[Object] in 0..1;
stages = [<X,weight>];
var int+ z;

maximize sum(i in Object) X[i]*value[i] - p*z
subject to{
z = max(0,expected(sum(i in Object) X[i]*weight[i] - c));
prob(sum(i in Object) X[i]*weight[i] - c ≤ 0) ≥ θ;
};
```

Stochastic OPL

DSKP

```
int N = 3;
int c = 10;
int p = 2;
float θ = 0.2
range Object [1..3];
int value[Object] = [8,15,10];
stoch int weight[Object] = [<5(0.5),8(0.5)>,
                      <3(0.5),9(0.5)>,<15(0.5),4(0.5)>];
var int+ X[Object] in 0..1;
stages = [<X[1],weight[1]>,<X[2],weight[2]>,<X[3],weight[3]>];
var int+ z;

maximize sum(i in Object) X[i]*value[i] - p*z
subject to{
z = max(0,expected(sum(i in Object) X[i]*weight[i] - c));
prob(sum(i in Object) X[i]*weight[i] - c ≤ 0) ≥ θ;
};
```

Scenario-based Compilation

By using the approach discussed in

S. A. Tarim, S. Manandhar and T. Walsh,
Stochastic Constraint Programming: A Scenario-Based Approach,
Constraints, Vol.11, pp.53-80, 2006

it is possible to compile any SCSP/SCOP down to a deterministic equivalent CSP.

Scenario-based Compilation

Stochastic Constraint Program

Objective:

$$\max \left\{ \sum_{i=1}^{k} r_i X_i - p\mathbb{E}\left[\sum_{i=1}^{k} \mathcal{W}_i X_i - c \right]^+ \right\}$$

Subject to:

$$\Pr\left\{ \sum_{i=1}^{k} \mathcal{W}_i X_i \leq c \right\} \geq \theta$$

Decision variables:

$X_i \in \{0,1\} \qquad \forall i \in 1,\ldots,k$

Stochastic variables:

$\mathcal{W}_i \rightarrow$ item i weight $\forall i \in 1,\ldots,k$

Stage structure:

$V_1 = \{X_1,\ldots,X_k\}$

$S_1 = \{\mathcal{W}_1,\ldots,\mathcal{W}_k\}$

$L = [(V_1, S_1)]$

Stochastic OPL Model

```
stoch myrand[onestage]=...;
int nbItems=...;
float c = ...;
float q = ...;
range Items 1..nbItems;
range onestage 1..1;
float W[Items,onestage]^myrand = ...;
float r[Items] = ...;
dvar float+ z;
dvar int x[Items] in 0..1;

maximize
sum(i in Items) x[i]*r[i] - expected(c*z)

subject to{
z >= sum(i in Items) W[i]*x[i] - q;
prob(sum(i in Items) W[i]*x[i] <= q) >= 0.6;
};
```

Compiler

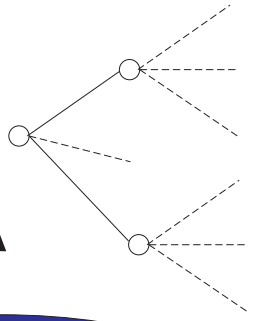
Deterministic equivalent model

```
int nbWorlds=...;
range Worlds 1..nbWorlds;
int nbItems=...;
range Items 1..nbItems;
float c = ...;
float W[Worlds,Items] =...;
float Pr[Worlds]=...;
float r[Items] = ...;
float q = ...;
var float+ z[Worlds];
var int+ x[Items] in 0..1;

maximize ((sum(i in Items)x[i]*r[i])-
          c*(sum(j in Worlds)Pr[j]*z[j]))
subject to{
   forall(j in Worlds)
       z[j]>=(sum(i in Items)W[j,i]*x[i])-q;
   sum(j in Worlds)
       Pr[j]*(sum(i in Items)W[j,i]*x[i] <= q) >= 0.2;
};
```

Solver

Solution



Scenario-based Compilation

SSKP: Compiled Deterministic Equivalent CSP

```
int nbWorlds=8;
range Worlds 1..nbWorlds;
int nbItems=3;
range Items 1..nbItems;
float p = 2;
float W[Worlds,Items] =[[5,3,15],
                        [5,3,4],
                        [5,9,15],
                        [5,9,4],
                        [8,3,15],
                        [8,3,4],
                        [8,9,15],
                        [8,9,4]
                        ];
float Pr[Worlds]=
[0.125,0.125,0.125,0.125,0.125,0.125,0.125,0.125];

float r[Items] = [8,15,10];
float c = 10;

var float+ z[Worlds];
var int+ w[Worlds] in 0..1;
var int+ x[Items] in 0..1;

maximize ((sum(i in Items)x[i]*r[i])-p*(sum(j in Worlds)Pr[j]*z[j]))

subject to{
   forall(j in Worlds) z[j]>=(sum(i in Items)W[j,i]*x[i])-c;
   forall(j in Worlds) (sum(i in Items)W[j,i]*x[i] <= c) => w[j]=1;
   sum(j in Worlds) Pr[j]*w[j] >= 0.2;
};
```

Scenario-based Compilation

Advantages

- **Seamless** Modeling under Uncertainty!
- **Stochastic OPL** not necessarily linked to CP

Drawbacks

- **Size** of the compiled model
- **Constraint Propagation** not fully supported

An Alternative Approach to Seamless Stochastic Optimization

Stochastic Constraint Program

Objective:

$$\max \left\{ \sum_{i=1}^{k} r_i X_i - p\mathbb{E}\left[\sum_{i=1}^{k} \mathcal{W}_i X_i - c \right]^+ \right\}$$

Subject to:

$$\Pr\left\{ \sum_{i=1}^{k} \mathcal{W}_i X_i \leq c \right\} \geq \theta$$

Decision variables:

$X_i \in \{0,1\} \qquad \forall i \in 1,\ldots,k$

Stochastic variables:

$\mathcal{W}_i \rightarrow$ item i weight $\forall i \in 1,\ldots,k$

Stage structure:

$V_1 = \{X_1,\ldots,X_k\}$

$S_1 = \{\mathcal{W}_1,\ldots,\mathcal{W}_k\}$

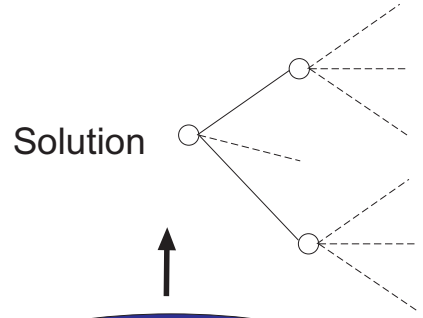
$L = [(V_1, S_1)]$

Stochastic OPL Model

```
stoch myrand[onestage]=...;
int nbItems=...;
float c = ...;
float q = ...;
range Items 1..nbItems;
range onestage 1..1;
float W[Items,onestage]^myrand = ...;
float r[Items] = ...;
dvar float+ z;
dvar int x[Items] in 0..1;

maximize
sum(i in Items) x[i]*r[i] - expected(c*z)

subject to{
z >= sum(i in Items) W[i]*x[i] - q;
prob(sum(i in Items) W[i]*x[i] <= q) >= 0.6;
};
```



Constraint Programming Solver supporting Global Chance-Constraints

Filtering Algorithms for Global Chance-Constraints

Filtering in SCSPs

Global Chance-Constraints

Perhaps the most interesting aspect of SCP is that the concept of **global constraint** can be also adopted in a stochastic environment, thus leading to

- Global Chance-Constraints (Rossi et al., 2008)

Stochastic Programming Model

$$\Pr\left\{\sum_{i=1}^{k} \mathcal{W}_i X_i \leq c\right\} \geq \theta$$

Global Chance-Constraint

```
stochLinIneq(x,W,Pr,c,0.2);
```

Filtering in SCSPs

Stochastic Constraint Programming

Global Chance-Constraints

- represent relations among a non-predefined number of **decision** and **random** variables
- implement dedicated filtering algorithms based on
  - **feasibility** reasoning
  - **optimality** reasoning

Global Chance-Constraints performing optimality reasoning are called **Optimization-Oriented Global Chance-Constraints** (Rossi et al., 2008).

Filtering in SCSPs

SSKP: Compiled Deterministic Equivalent CSP with Global Chance-Constraints

```
int nbWorlds=8;
range Worlds 1..nbWorlds;
int nbItems=3;
range Items 1..nbItems;
float c = 2;
float W[Worlds,Items] =[[5,3,15],
                        [5,3,4],
                        [5,9,15],
                        [5,9,4],
                        [8,3,15],
                        [8,3,4],
                        [8,9,15],
                        [8,9,4]
                        ];
float Pr[Worlds]=
[0.125,0.125,0.125,0.125,0.125,0.125,0.125,0.125];

float r[Items] = [8,15,10];
float q = 10;

var float+ z;
var int+ x[Items] in 0..1;

maximize ((sum(i in Items)x[i]*r[i])-c*(max(0,z-q)))

subject to{
   stochLinIneq(x,W,Pr,q,0.2);
   expectedLinEq(x,W,Pr,z);
};
```

Filtering in SCSPs

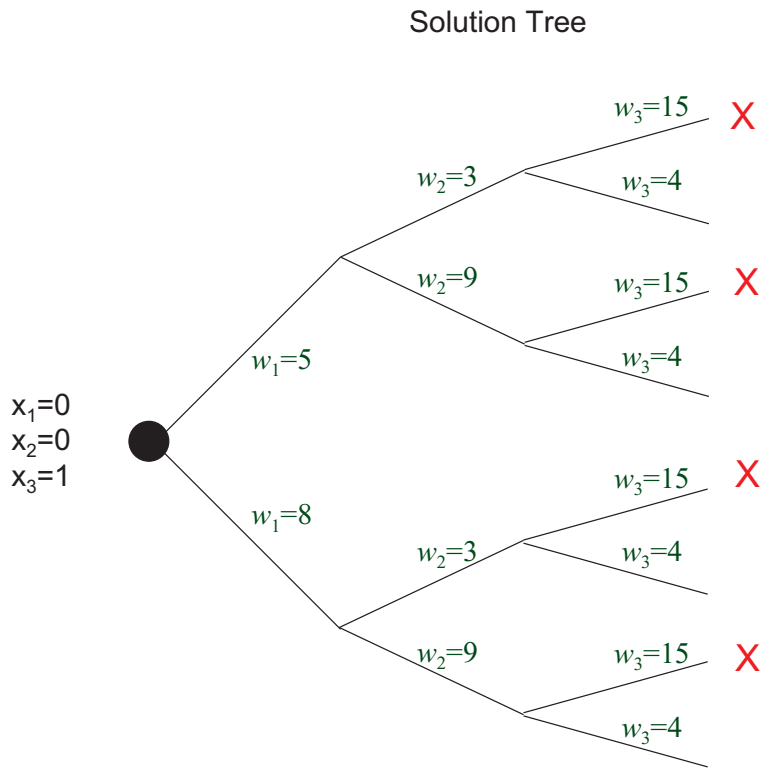
Algorithm 1: Filtering Algorithm

input : h; $\mathcal{PT}$; $\mathcal{A}$.
output: Filtered $\mathcal{PT}$ wrt h.

```
1  begin
2      for each i ∈ {0,...,N − 1} do
3          for each v ∈ D(PT[i]) do
4              f[i, v] ← 0;
5      for each p ∈ Ψ do
6          Create a copy c of h_{↓p} and of the decision variables it constrains;
7          Enforce GAC on c using A;
8          for each index i of the variables in c do
9              for each v in domain of the copy of PT[i] do
10                 f[i, v] ← f[i, v] + Pr{arcs(p)};
11         delete c and the respective copies of the decision variables;
12     for each i ∈ {0,...,N − 1} do
13         max[i] ← 0;
14         for each v ∈ D(PT[i]) do
15             max[i] ← max(max[i], f[i, v]);
16     for each k ∈ {1,...,m} do
17         g[k] ← 0;
18         for each i ∈ M_k do
19             g[k] ← g[k] + max[i]
20     for each k ∈ {1,...,m} do
21         for each i ∈ M_k do
22             for each v ∈ PT[i] do
23                 if g[k] − max[i] + f[i, v] < θ_k then
24                     prune value v from D(PT[i]);
25 end
```

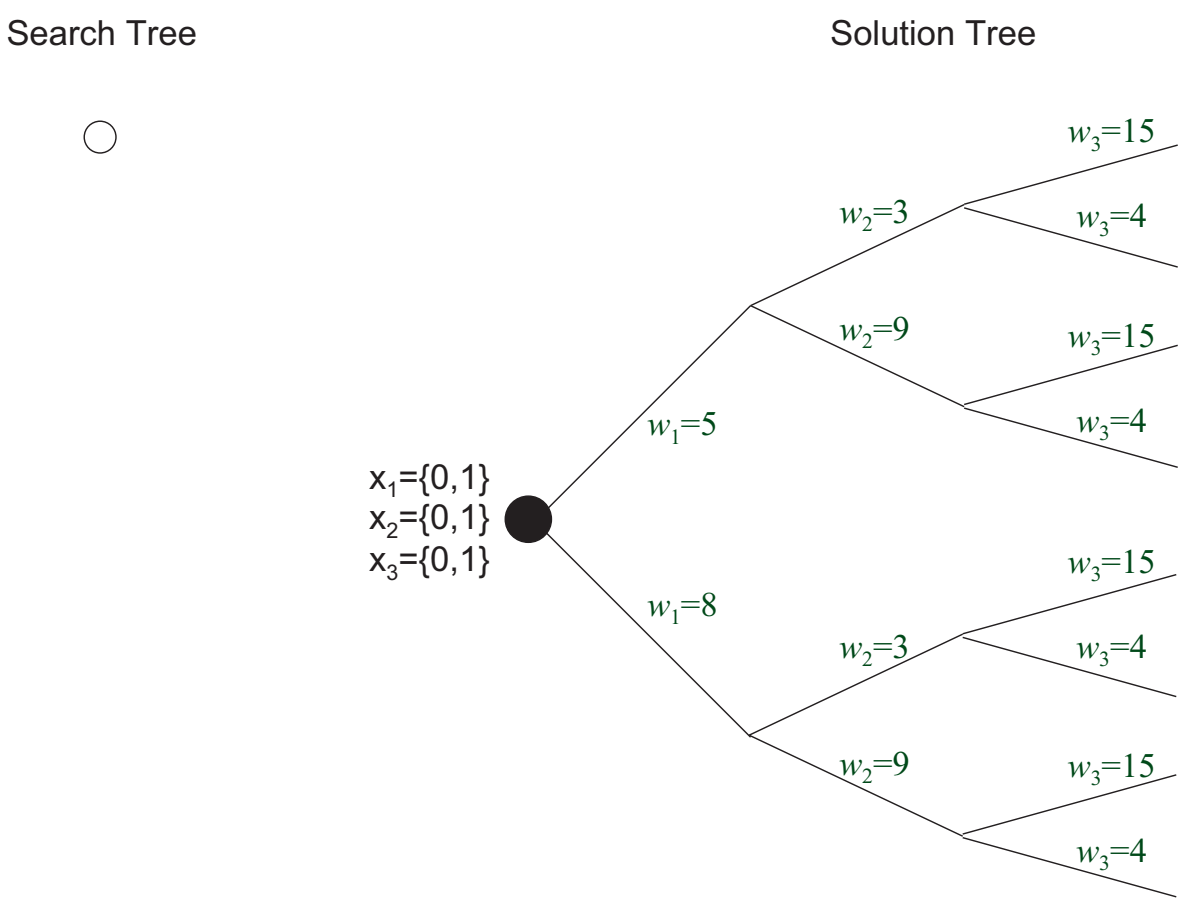
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$



Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$

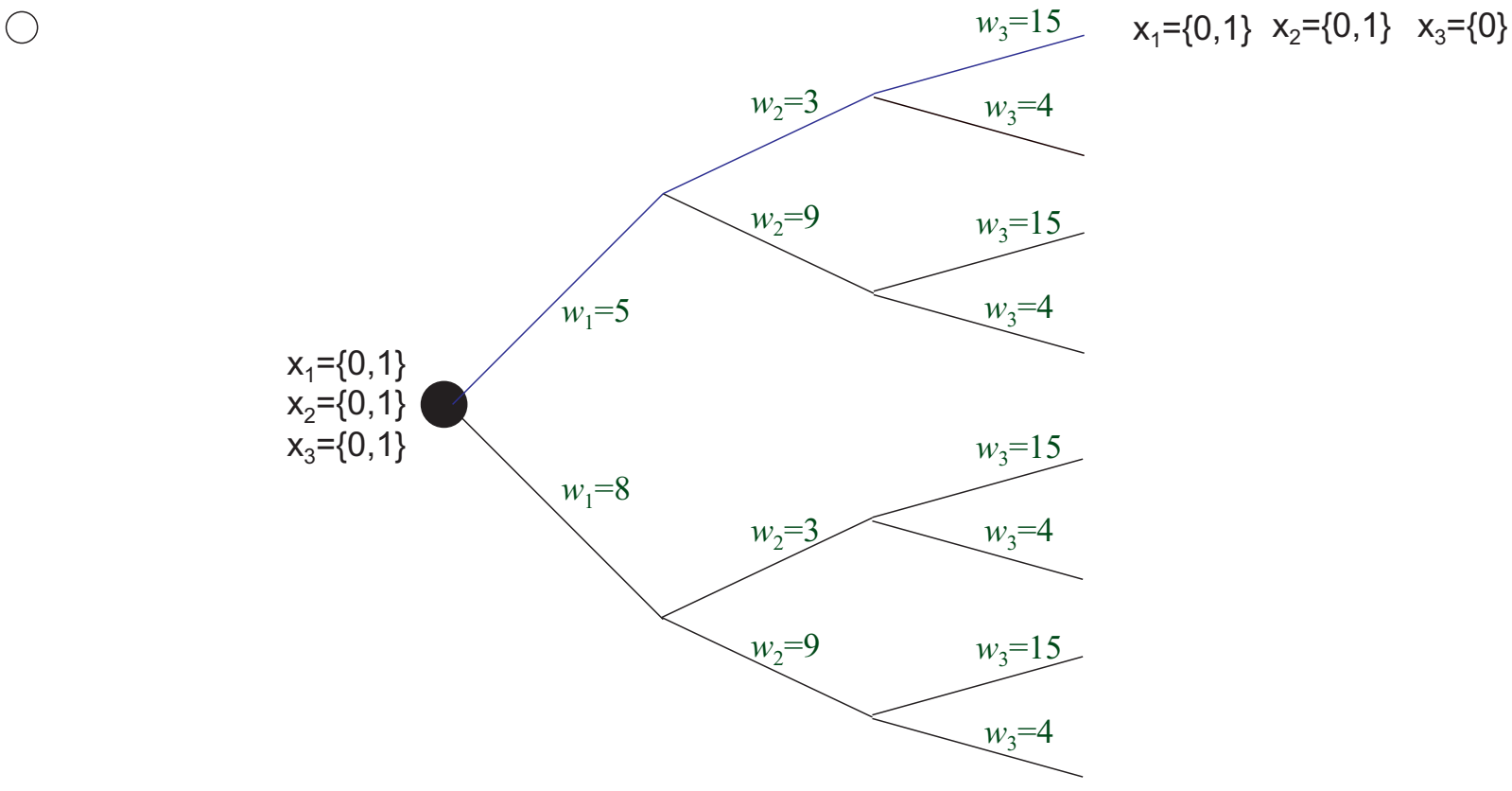


Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$

Search Tree

Solution Tree

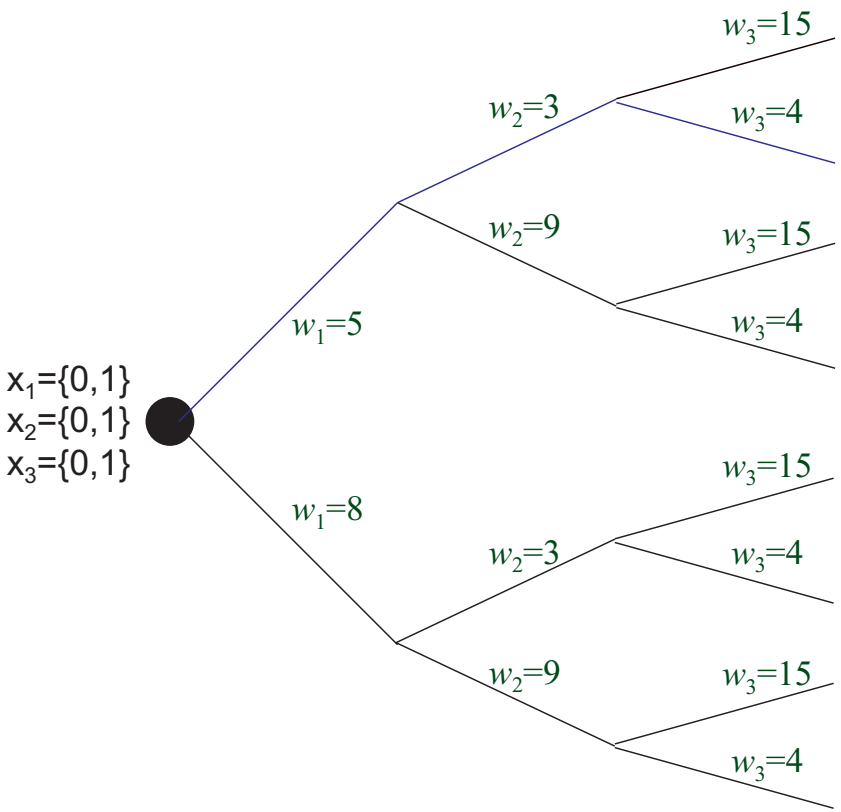


Filtering Algorithms for GCCs: An example

$Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$

Search Tree

Solution Tree



$x_1=\{0,1\}$ $x_2=\{0,1\}$ $x_3=\{0\}$

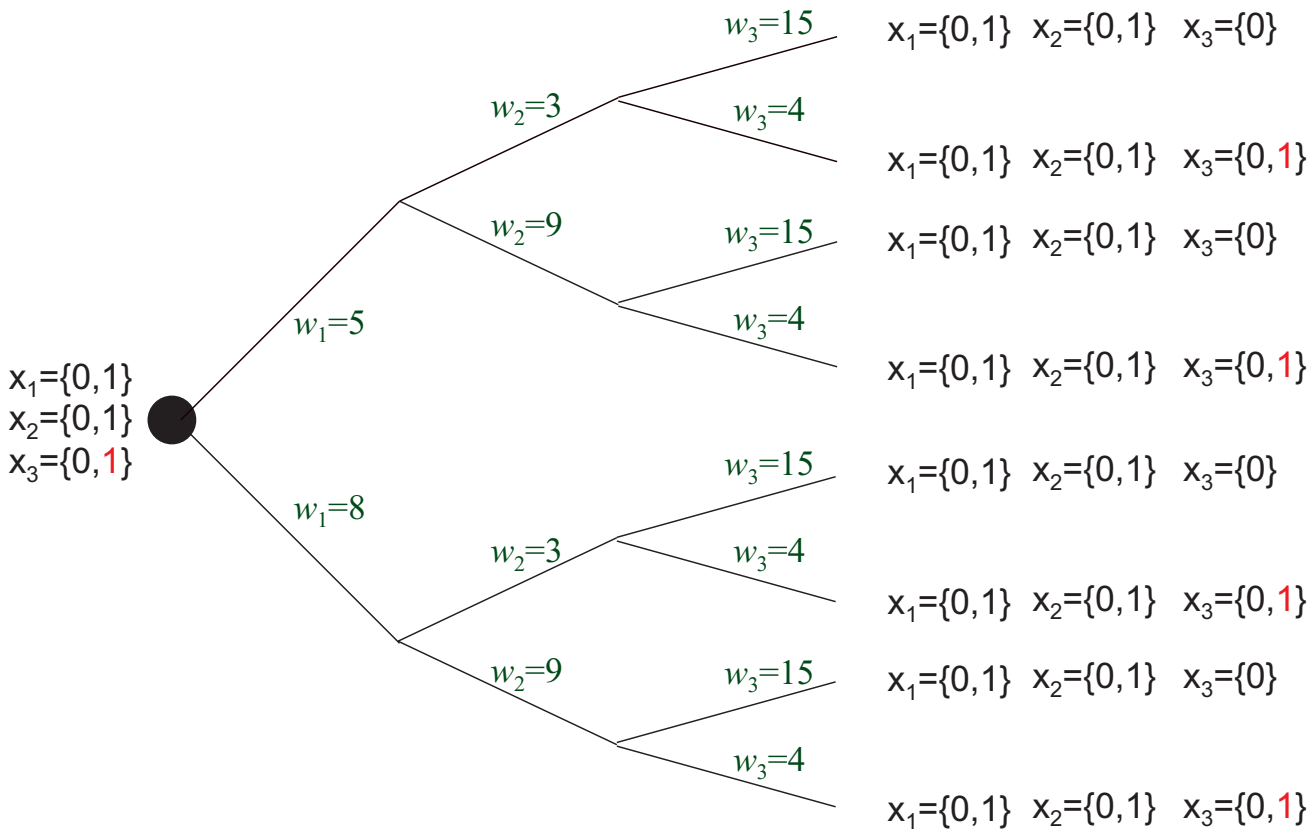
$x_1=\{0,1\}$ $x_2=\{0,1\}$ $x_3=\{0,1\}$

Filtering Algorithms for GCCs: An example

$Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$

Search Tree

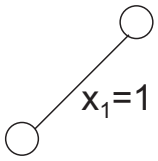
Solution Tree



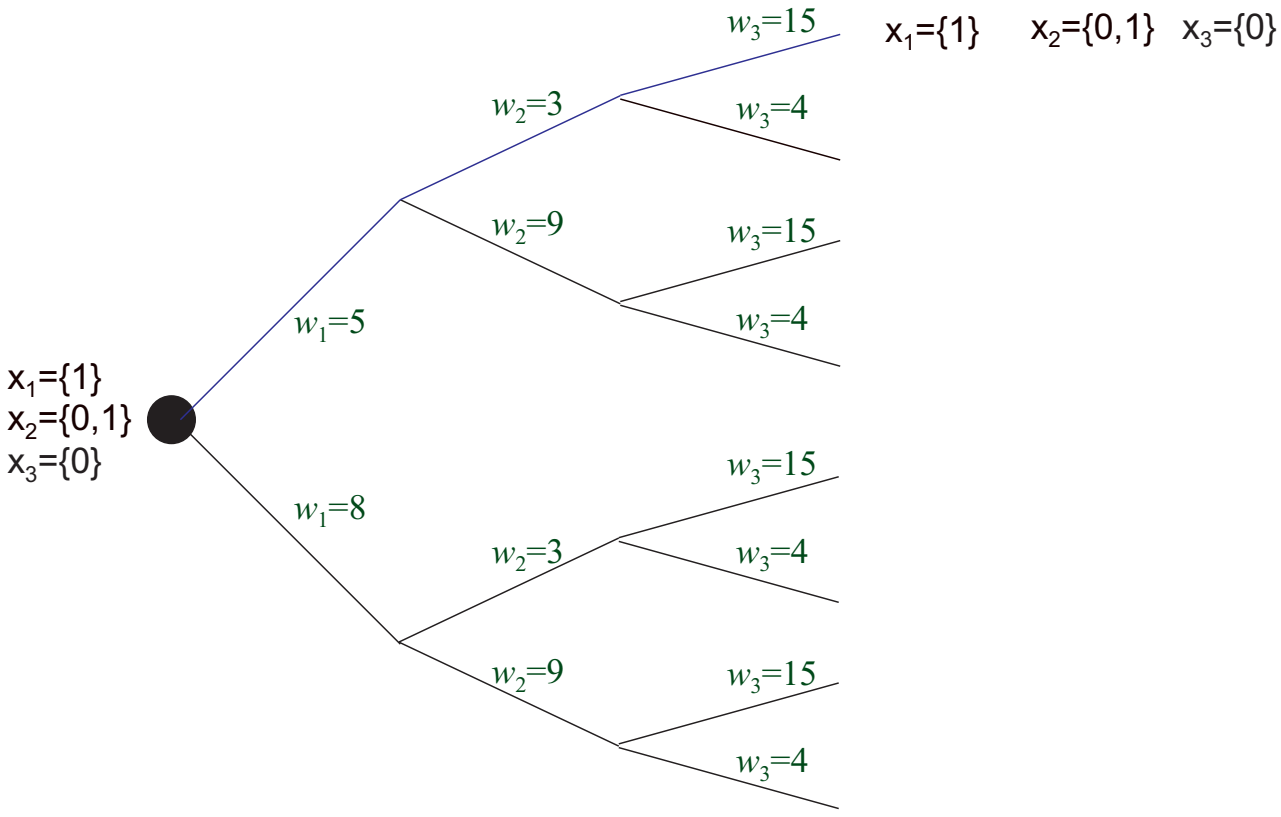
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$

Search Tree



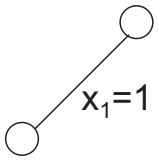
Solution Tree



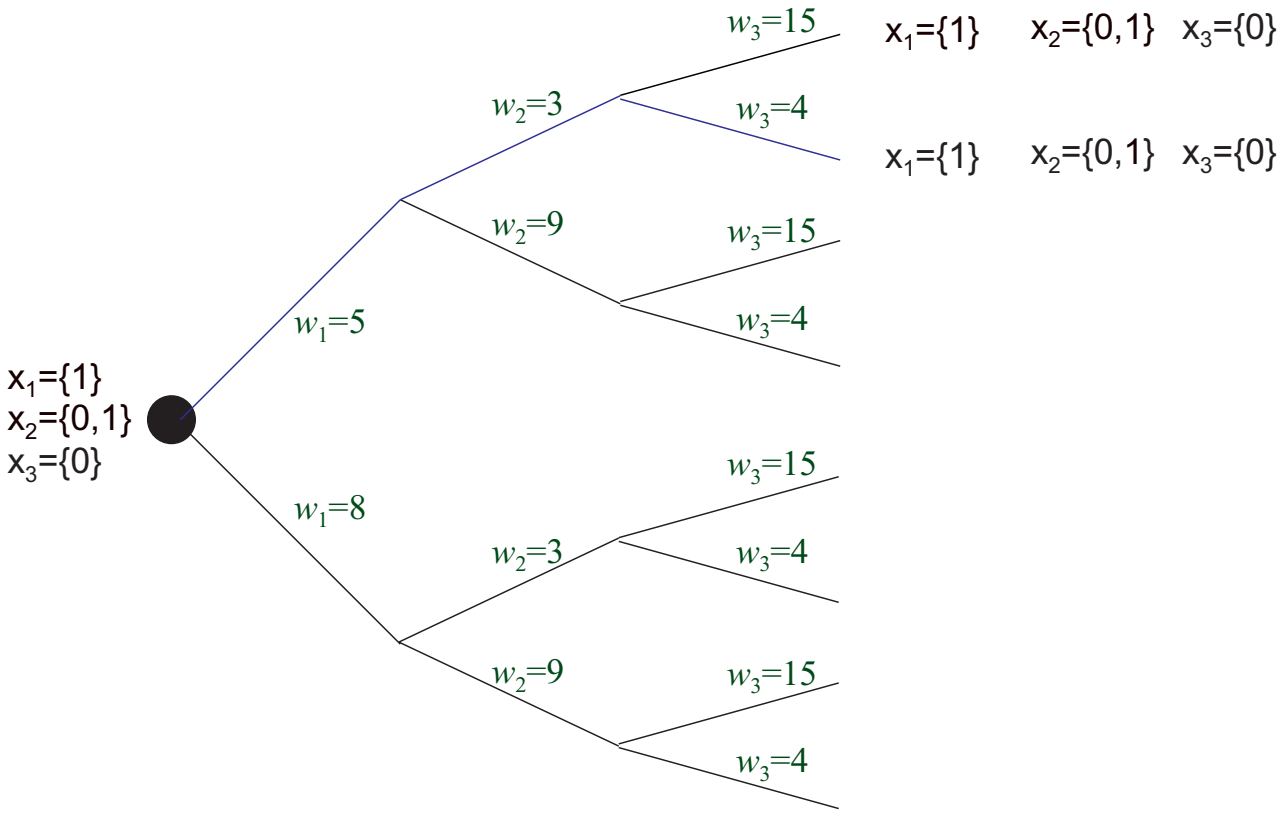
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$

Search Tree



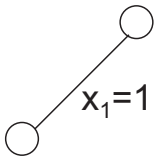
Solution Tree



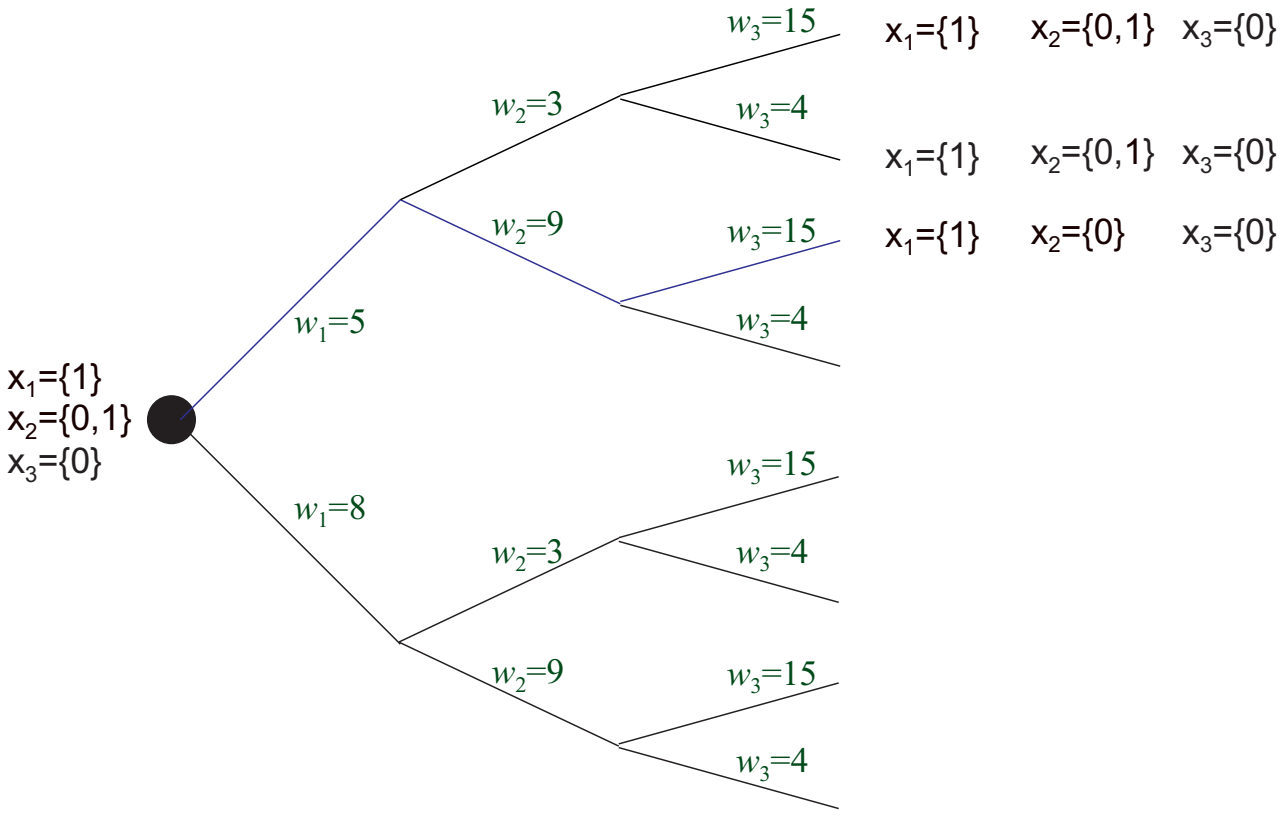
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$

Search Tree



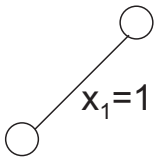
Solution Tree



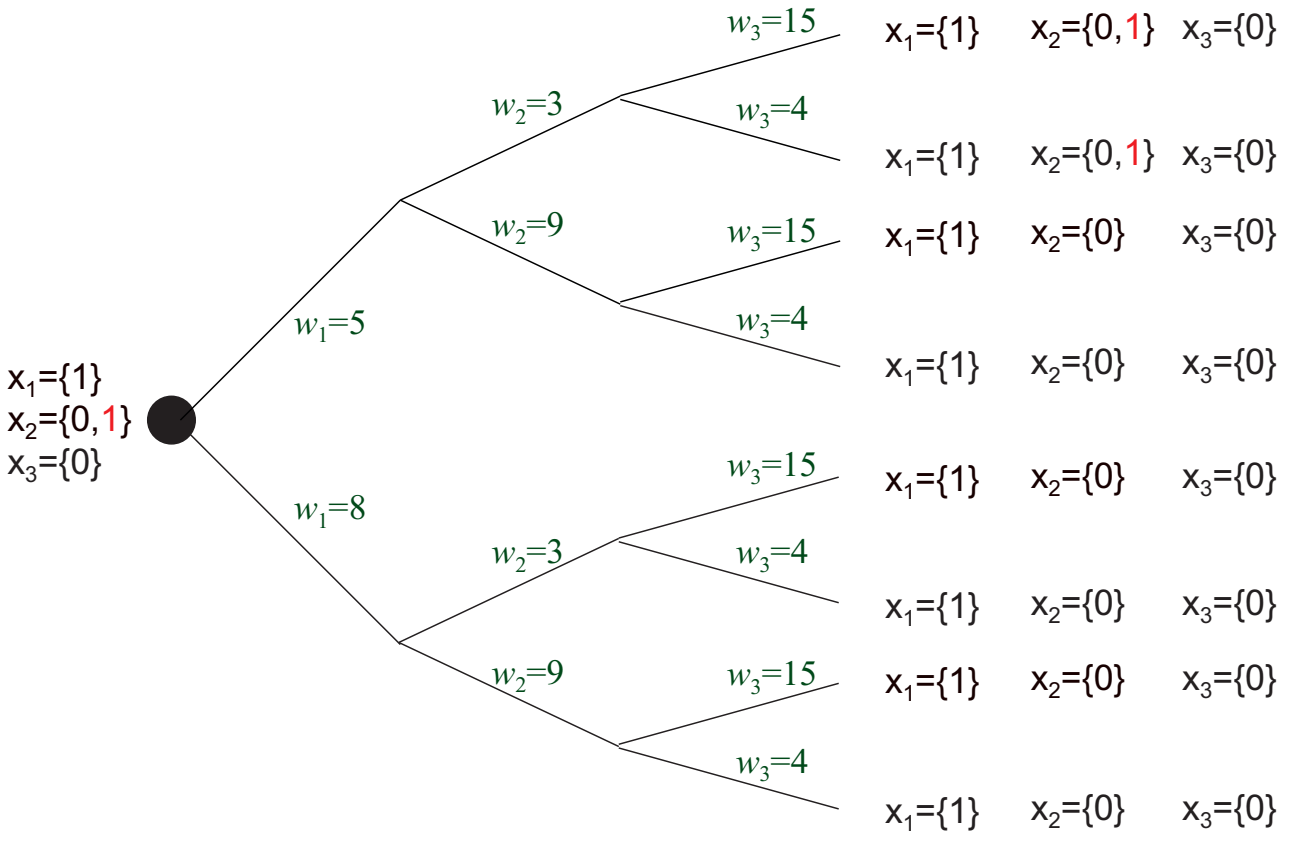
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$

Search Tree



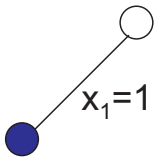
Solution Tree



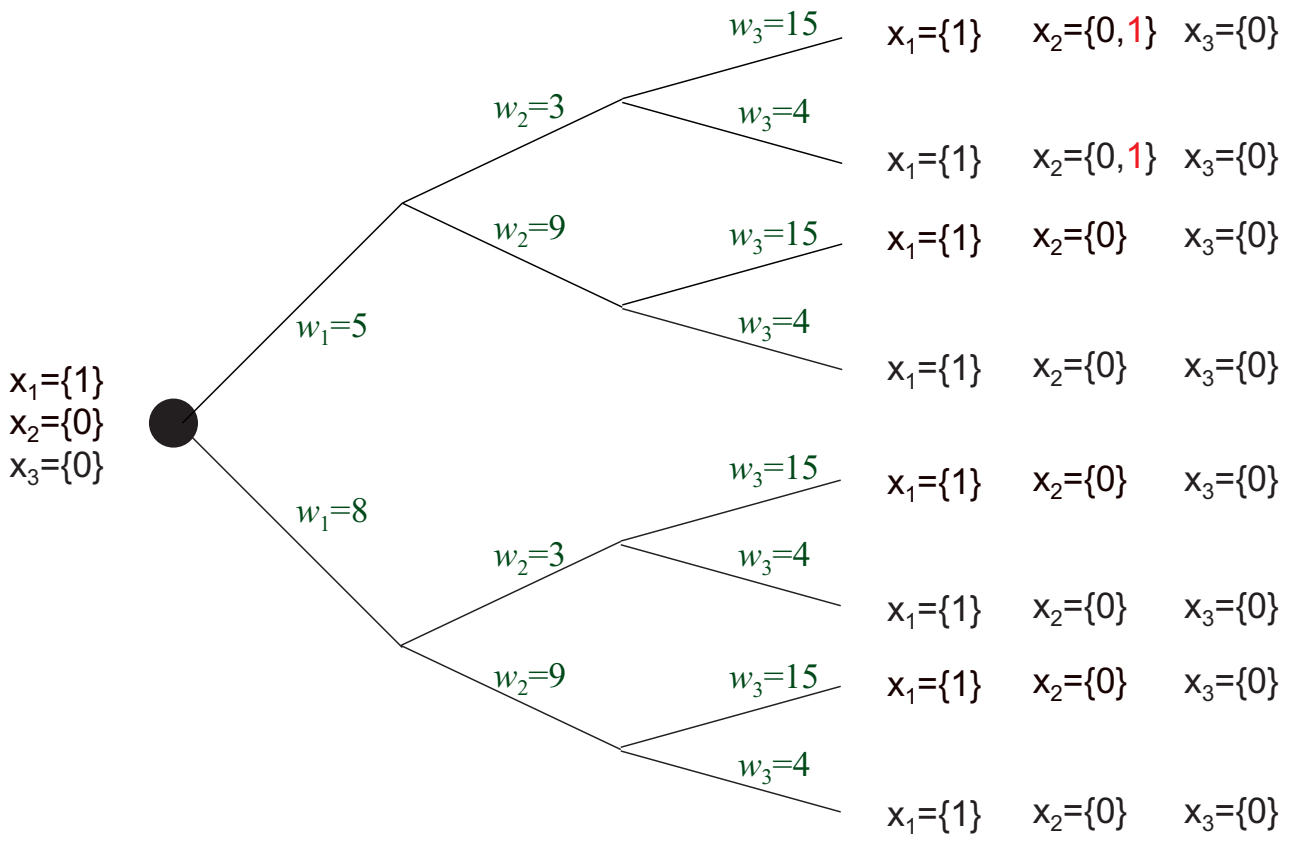
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) > 0.5$

Search Tree



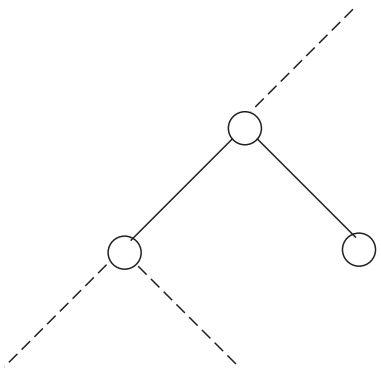
Solution Tree



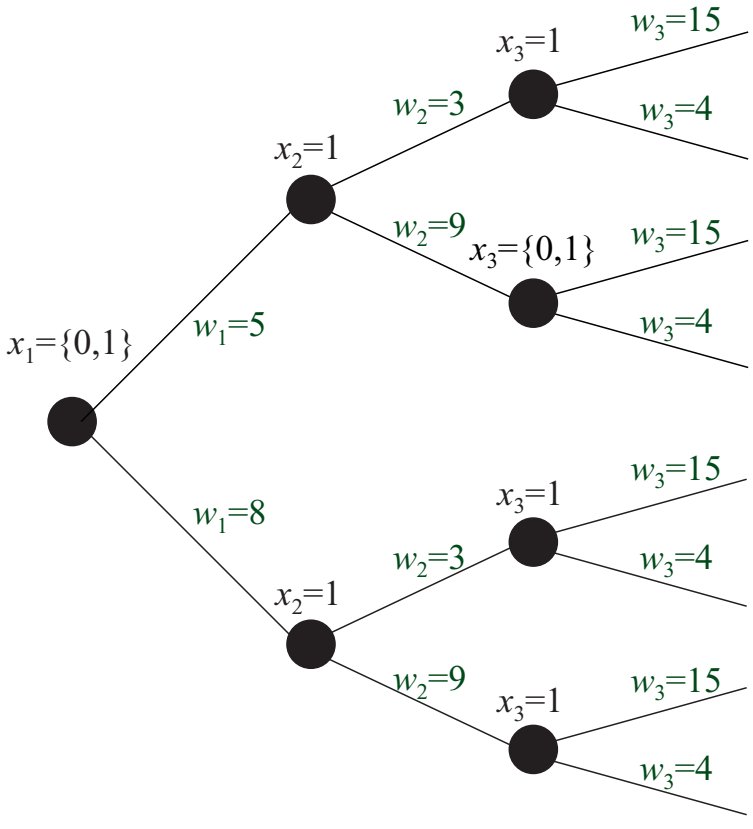
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



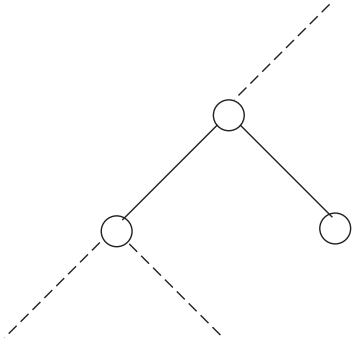
Solution Tree



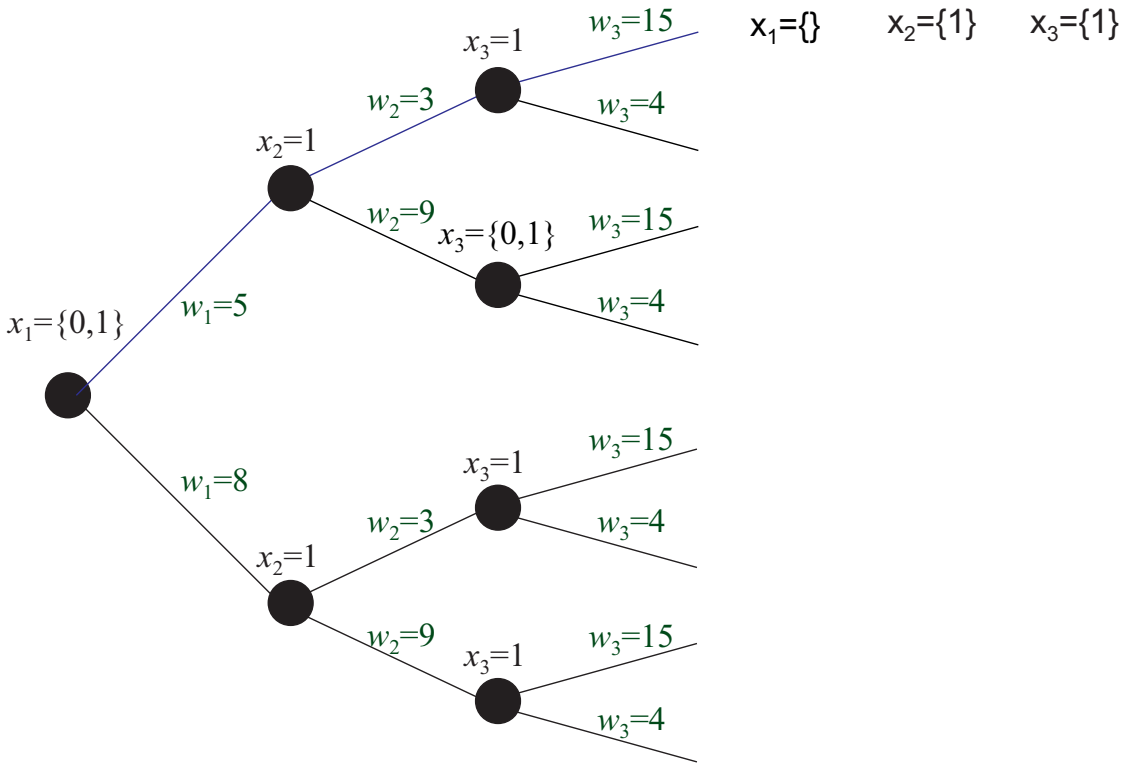
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



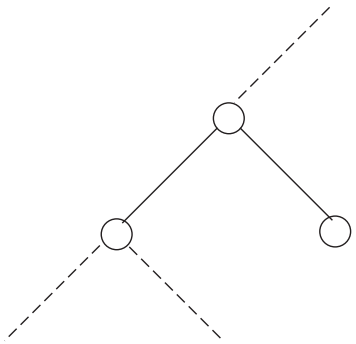
Solution Tree



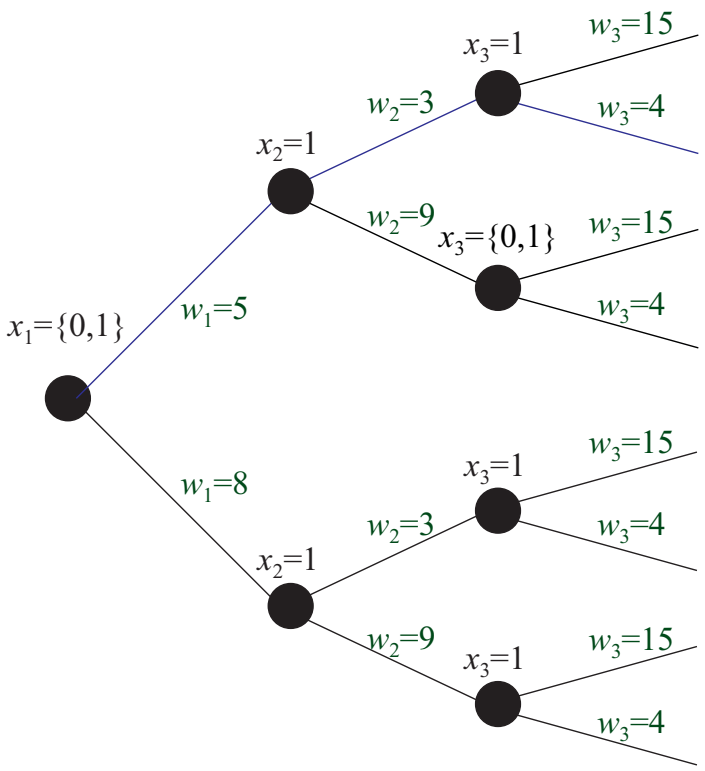
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



Solution Tree



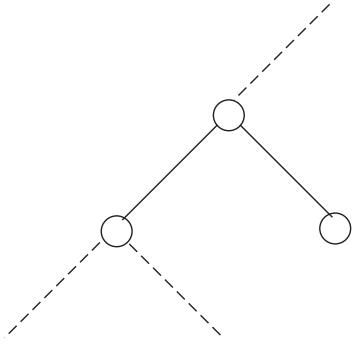
x_1={} x_2={1} x_3={1}

x_1={0} x_2={1} x_3={1}

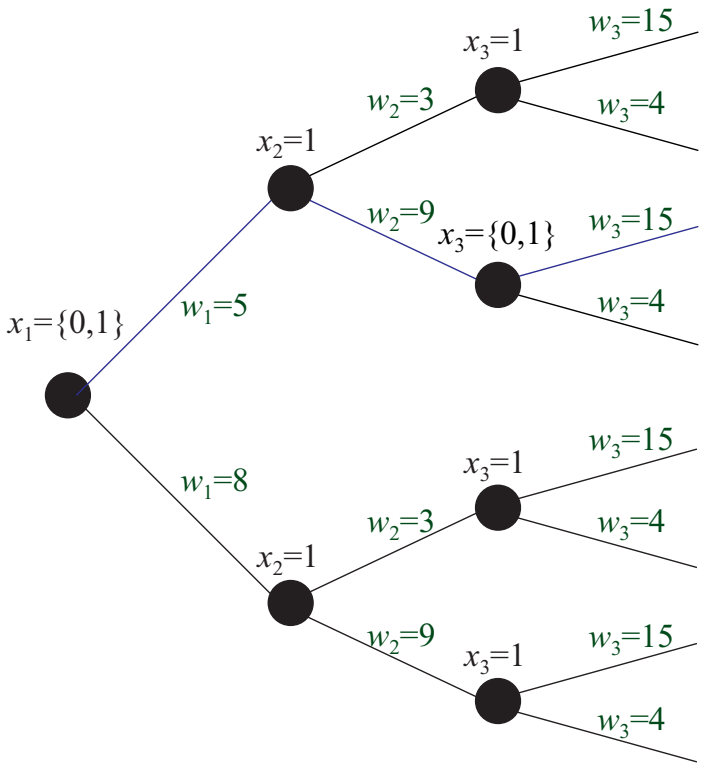
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



Solution Tree



x_1={} x_2={1} x_3={1}

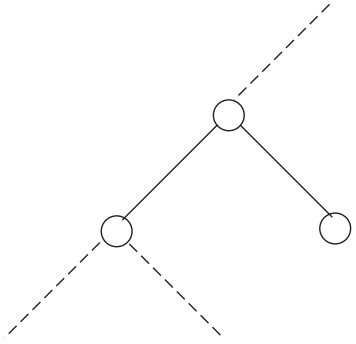
x_1={0} x_2={1} x_3={1}

x_1={0} x_2={1} x_3={0}

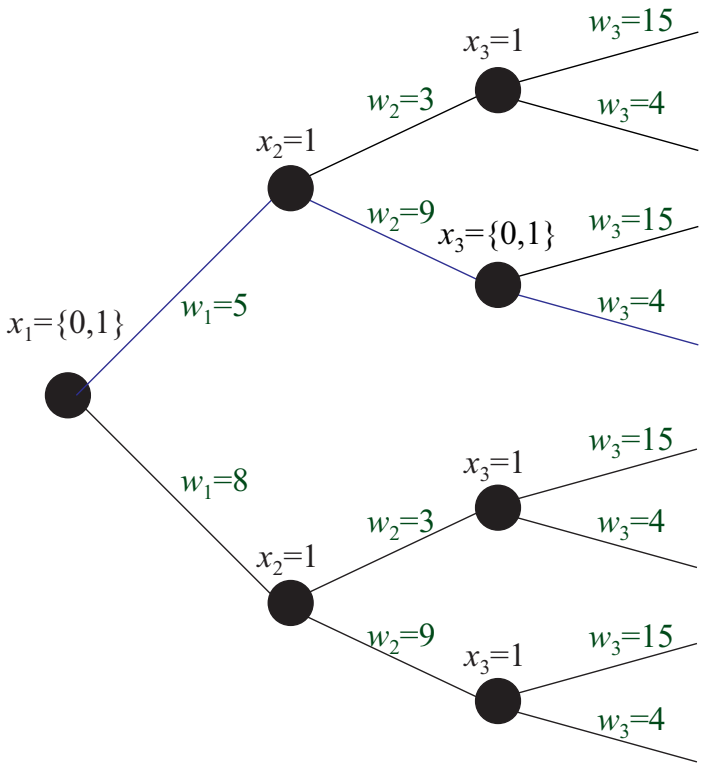
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



Solution Tree



x_1={} x_2={1} x_3={1}

x_1={0} x_2={1} x_3={1}

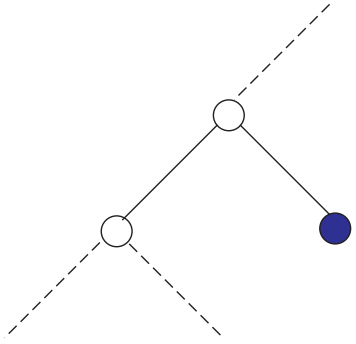
x_1={0} x_2={1} x_3={0}

x_1={0} x_2={1} x_3={0}

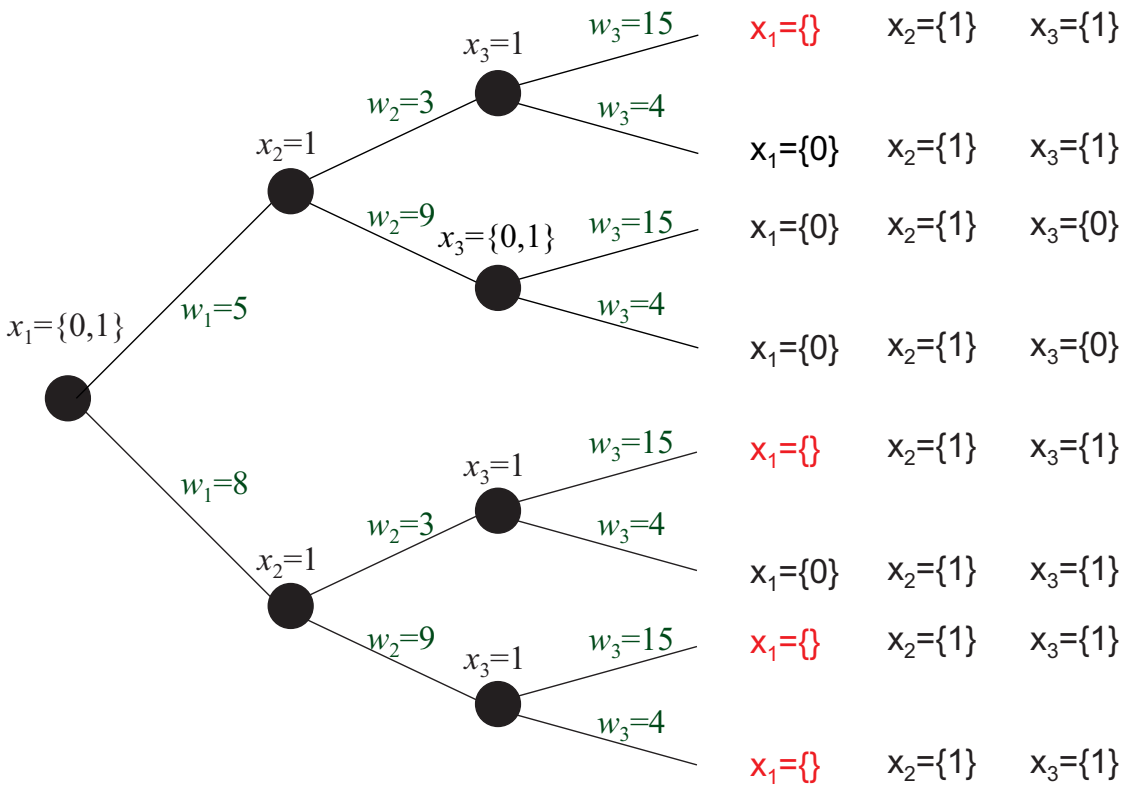
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



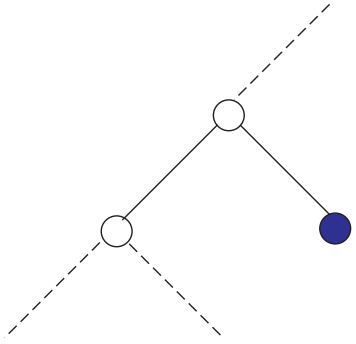
Solution Tree



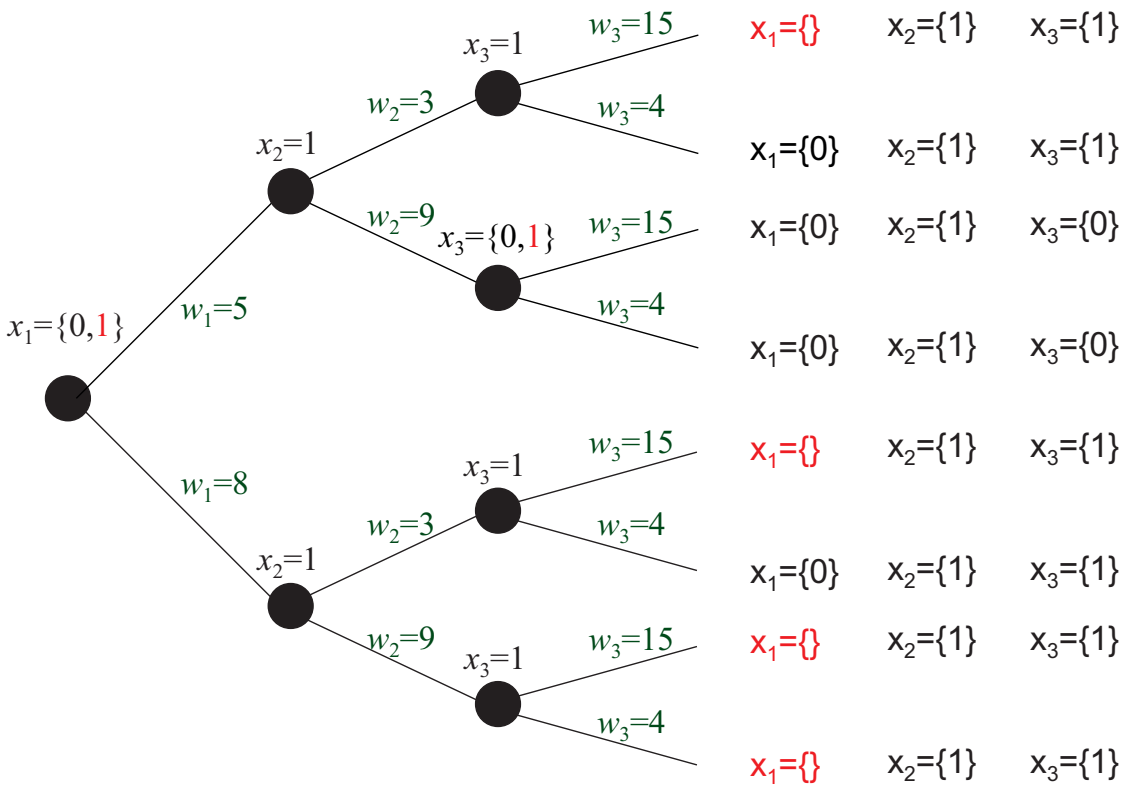
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



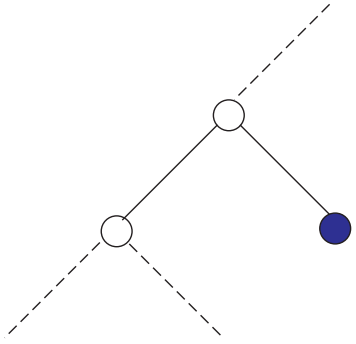
Solution Tree



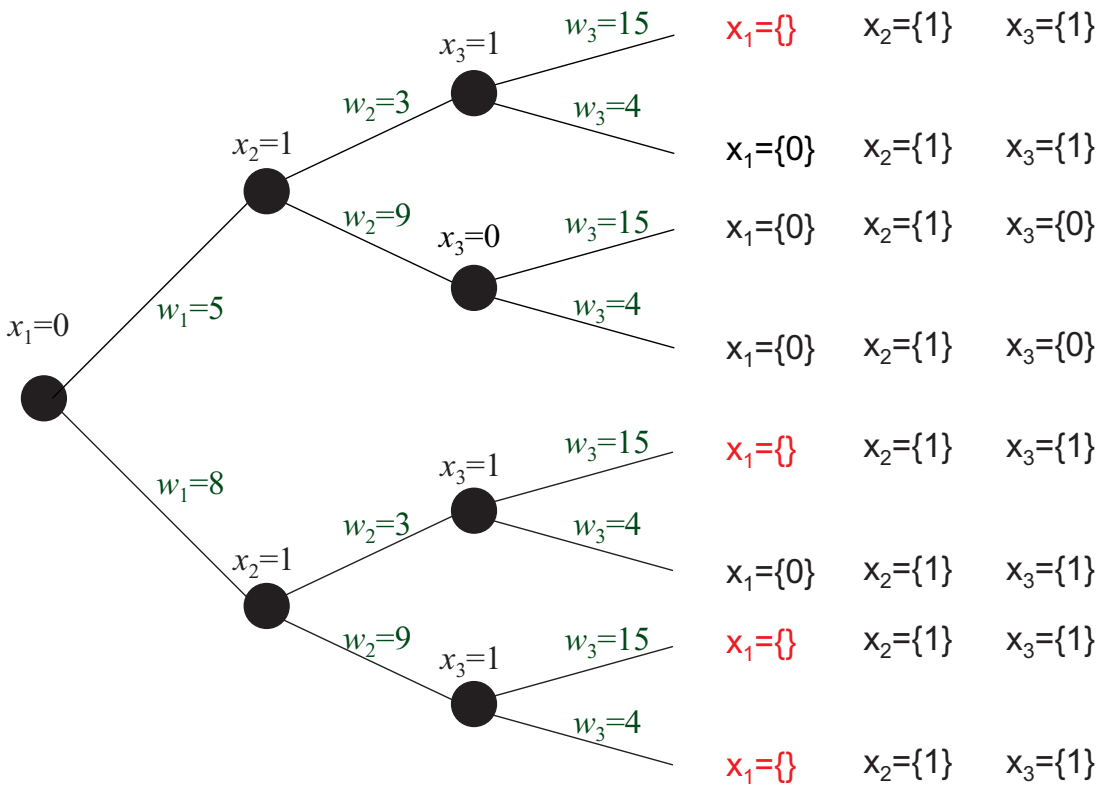
Filtering Algorithms for GCCs: An example

$\Pr(w_1x_1+w_2x_2+w_3x_3 \leq 10) \geq 0.5$

Search Tree



Solution Tree



Random SCSPs

- In our experiments we considered a number of **randomly generated SCSPs**
- The SCSPs considered feature
  - 5 chance-constraints
  - 4 integer decision variables, $x_1, \ldots, x_4$
  - 8 stochastic variables, $s_1, \ldots, s_8$
  - 3 possible stage structure (single and multi-stage problems)

Model Size

- The model that uses GCC is much more compact!

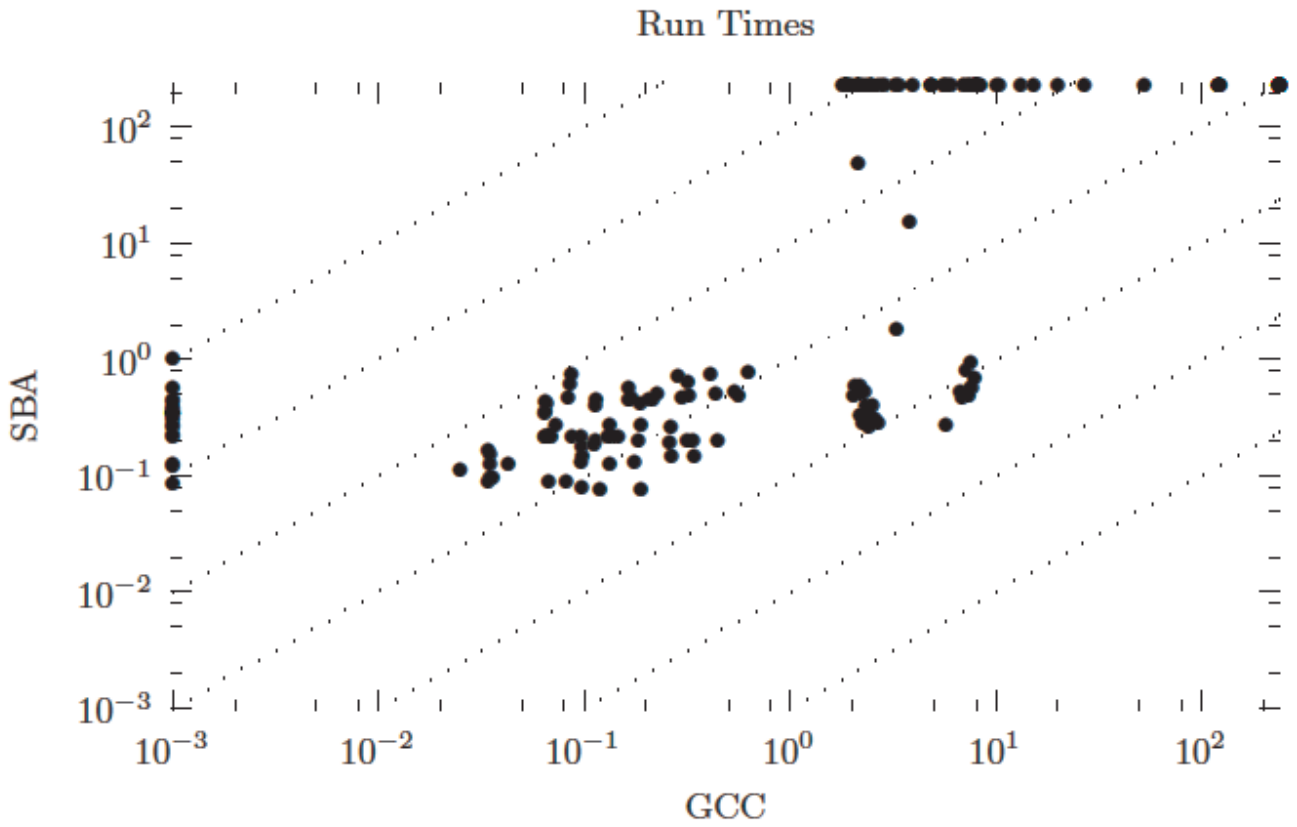
	SBA		GCC	
Stages	Dec Vars	Cons	Dec Vars	Cons
1	6484	6485	4	5
2	6554	6485	74	5
4	6739	6485	259	5

Results

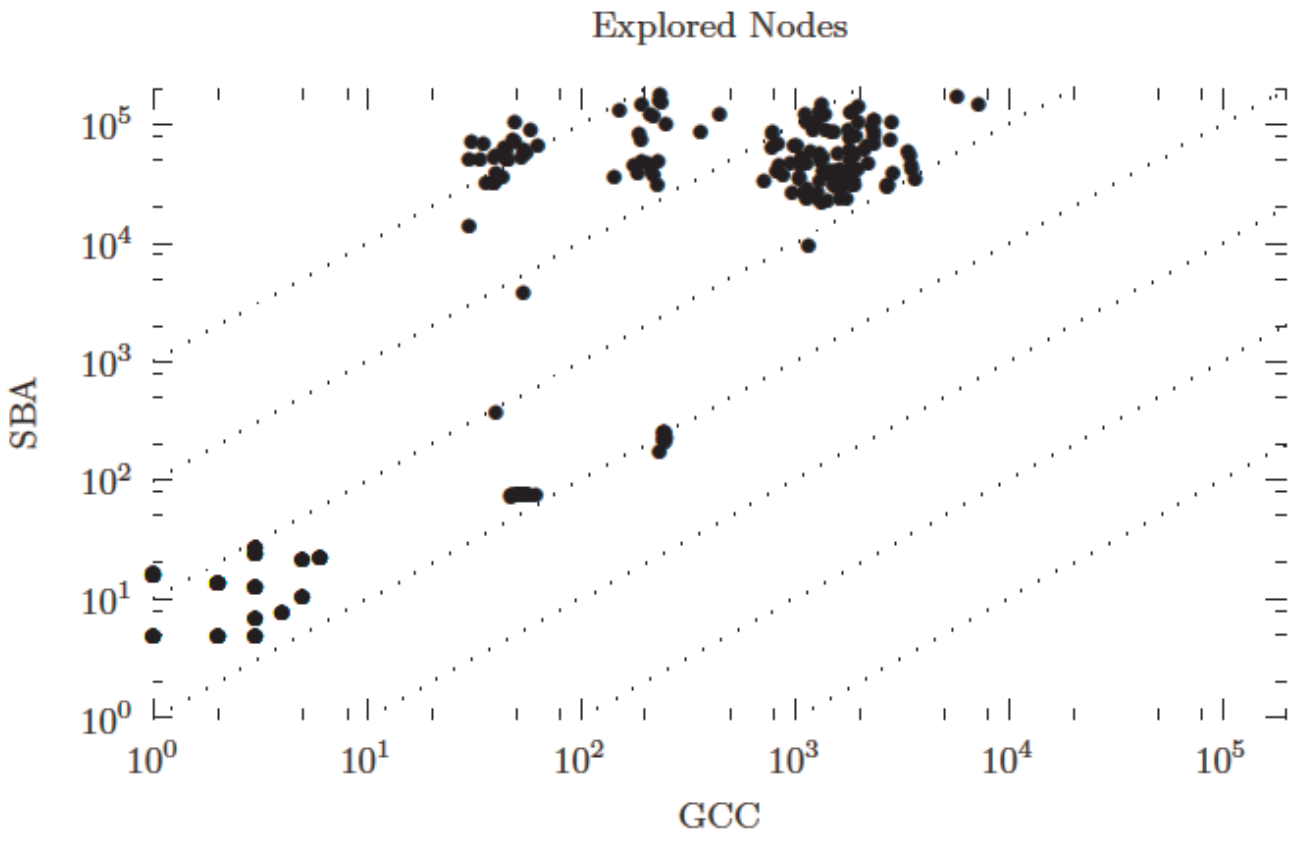
- The test bed comprised, in total, 270 instances
- To each instance we assigned a time limit of 240 seconds for running the search

Stages	Solved Instances SBA	GCC	Speed up GCC	Node Gain GCC
1	90	90	2.5×	5×
2	18	45	13×	400×
4	10	31	15×	300×

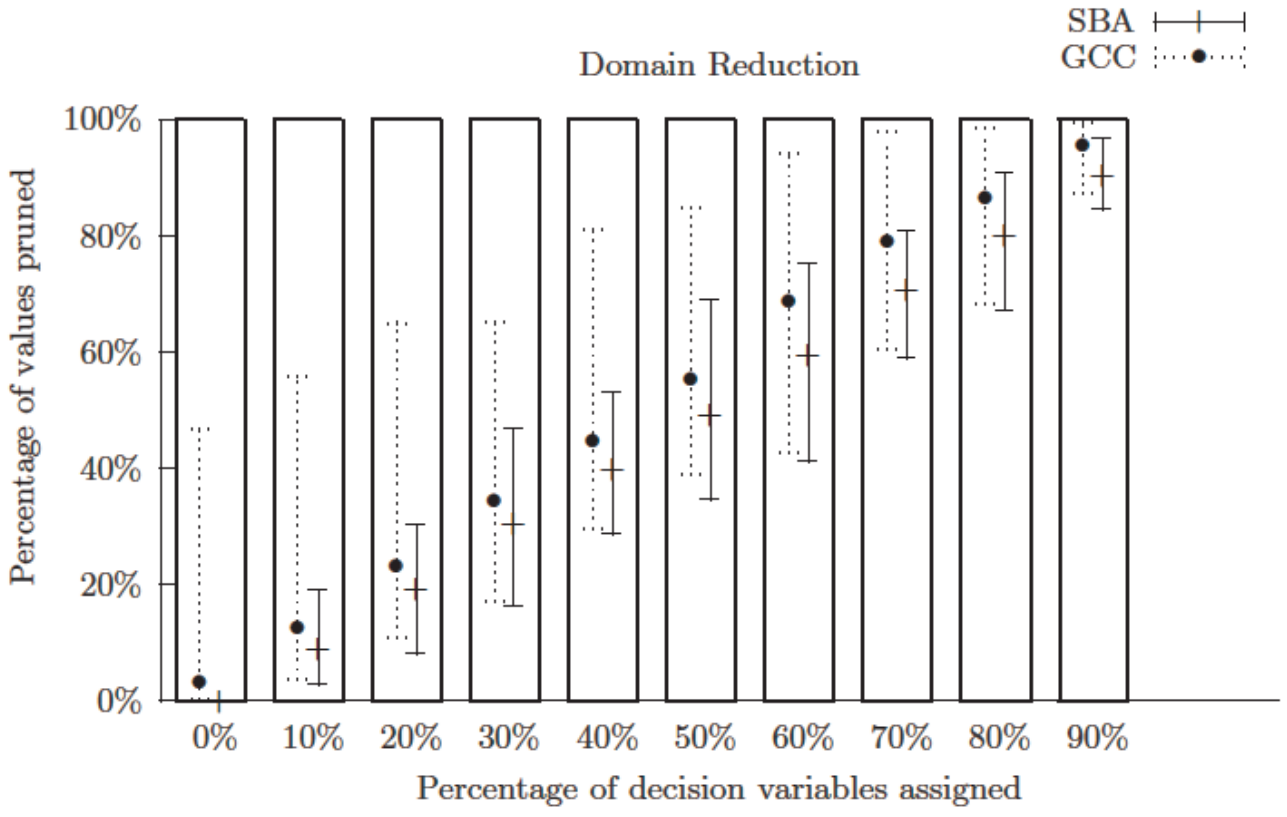
Run Times


Run Times
SBA
GCC
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}

Explored Nodes



Filtering


SBA
GCC
Domain Reduction
Percentage of values pruned
100%
80%
60%
40%
20%
0%
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
Percentage of decision variables assigned

Summary

We discussed a **Framework** for **Modeling Decision Problems under Uncertainty**

- Stochastic Constraint Programming
- Global Chance-constraints

Contribution

A generic approach for **constraint reasoning under uncertainty**.
Works with any existing propagation algorithm!

Drawback

Only implemented for linear inequalities/equalities:
i.e. SSKP $\rightarrow$ $\Pr(w_1x_1 + w_2x_2 + w_3x_3 \leq 10) \geq 0.2$

Questions