Synthesizing Filtering Algorithms for Global Chance-Constraints

Brahim Hnich\textsuperscript{1}
Roberto Rossi\textsuperscript{2}
S. Armagan Tarim\textsuperscript{3}
Steven D. Prestwich\textsuperscript{4}

\textsuperscript{1}Faculty of Computer Science, Izmir University of Economics, Izmir, Turkey
\textsuperscript{2}LDI, Wageningen UR, the Netherlands
\textsuperscript{3}Dept. of Operations Management, Nottingham University Business School, UK
\textsuperscript{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

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 $\theta$.

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
- $\theta_h$ 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 $f(\hat{V}, \hat{S})$ we obtain a SCOP.
An Example

Sample SCOP: SSKP

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

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

maximize sum(i in Object) X[i] \cdot value[i] - p \cdot z
subject to{
   z = \max(0,\text{expected}(\text{sum}(i \in Object) X[i] \cdot \text{weight}[i] - c));
   \text{prob}(\text{sum}(i \in Object) X[i] \cdot \text{weight}[i] - c \leq 0) \geq \theta;
};
Scenario-based Compilation

By using the approach discussed in


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

Stochastic Constraint Programming

**Solution Methods**

**Scenario-based Compilation**

**Stochastic Constraint Program**

Objective:

max \( \left\{ \sum_{i=1}^{n} r_i X_i - p \left[ \sum_{i=1}^{n} W_i X_i - c \right] \right\} \)

Subject to:

\[ \begin{align*}
& \text{stoch myrand(onestage)=...;}
& \text{int nbItems=...;}
& \text{float c = ...;}
& \text{float q = ...;}
& \text{range Items 1..nbItems;}
& \text{range onestage 1..2;}
& \text{float W[Items,onestage]*myrand = ...;}
& \text{float r[Items] = ...;}
& \text{dvar float z = ...;}
& \text{dvar int x[Items] in 0..1;}
& \text{maximize}
& \text{sum(i in Items) x[i]*r[i] - expected(c*q)}
& \text{subject to:}
& \text{z := sum(i in Items) W[i]*x[i] = q;}
& \text{prob(sum(i in Items) W[i]*x[i] <= q) >= 0.6;}
\end{align*} \]

**Stochastic OPL Model**

```opl
stoch myrand(onestage)=...;
int nbItems=...;
float c = ...;
float q = ...;
range Items 1..nbItems;
rangepos onestage 1..2;
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] - c*sum(i in Items) Pr[i]*x[i])]
subject to:
forall(i in Items)
  x[i]=sum(i in Items) W[i]*x[i] = q;
prob(sum(i in Items) Pr[i]*x[i] <= q) >= 0.6;
```

**Deterministic equivalent model**

```opl
int nbWorlds=...;
rangepos Worlds 1..nbWorlds;
int nbItems=...;
rangepos Items 1..nbItems;
float c = ...;
float q = ...;
rangepos Worlds,Items 1..2;
float Pr[Worlds]=...;
rangepos Items 1..2;
dvar float z = ...;
dvar int x[Items] in 0..1;
maximize 
[sum(i in Items) x[i]*r[i] - c*sum(i in Worlds) Pr[i]*x[i])]
subject to:
forall(i in Worlds)
  x[i]=sum(i in Items) W[i]*x[i] = q;
prob(sum(i in Worlds) Pr[i]*x[i] <= q) >= 0.6;
```

**Compiler**

**Solver**

**Solution**
SSKP: Compiled Deterministic Equivalent CSP

```java
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:
\[ \max \{ \sum_{i=1}^{n} r_i x_i - p \left( \sum_{i=1}^{n} w_i x_i - c \right) \} \]

Subject to:
\[ \Pr \left\{ \sum_{i=1}^{n} w_i x_i \leq c \right\} \geq \theta \]

Decision variables:
\[ x_i \in \{0, 1\}, \quad i = 1, \ldots, k \]

Stage structure:
\[ v_1 = \{ x_1, \ldots, x_k \}, \quad s_1 = (w_1, \ldots, w_k), \quad L = [v_1, s_1] \]

Stochastic OPL Model

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

Solution Methods

Constraint Programming Solver supporting Global Chance-Constraints

Filtering Algorithms for Global Chance-Constraints
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

\[ \Pr \left\{ \sum_{i=1}^{k} W_i X_i \leq c \right\} \geq \theta \]

Global Chance-Constraint

\texttt{stochLinIneq}(x,W,Pr,c,0.2);
Global Chance-Constraints

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).
SSKP: Compiled Deterministic Equivalent CSP with Global Chance-Constraints

```c
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],
                        [8.9,4],
                        [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{
    stochasticLinIneq(x,W,Pr,q,0.2);
    expectedLinEq(x,W,Pr,z);
}
```
Global Chance-Constraints

Filtering in SCSPs

Algorithm 1: Filtering Algorithm

```
Algorithm 1: Filtering Algorithm
input: h \in \mathcal{PT}; A
output: Filtered \mathcal{PT} wrt h
1 begin
2     for each i \in \{0, \ldots, N-1\} do
3         for each v \in D(\mathcal{PT}[i]) do
4             \checkmark f[i,v] \leftarrow 0;
5     for each p \in \Phi 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 \mathcal{PT}[i] do
10                f[i,v] \leftarrow f[i,v] + \text{Pr}(\text{arcs}(p));
11         delete c and the respective copies of the decision variables;
12     for each i \in \{0, \ldots, N-1\} do
13         max[i] \leftarrow 0;
14         for each v \in D(\mathcal{PT}[i]) do
15             max[i] \leftarrow \max\{max[i], f[i,v]\};
16     for each k \in \{1, \ldots, m\} do
17         g[k] \leftarrow 0;
18         for each i \in M_k do
19             g[k] \leftarrow g[k] + max[i]
20     for each k \in \{1, \ldots, m\} do
21         for each l \in M_k do
22             for each v \in D(\mathcal{PT}[i]) do
23                 if g[k] - max[i] + f[i,v] < \theta, then
24                     prune value v from D(\mathcal{PT}[i]);
25 end
```
Global Chance-Constraints

Filtering Algorithms for GCCs: An example

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

Solution Tree

- \( w_1 = 5 \)
- \( w_2 = 3 \)
- \( w_3 = 4 \)
- \( x_1 = 0 \)
- \( x_2 = 0 \)
- \( x_3 = 1 \)

- \( w_1 = 8 \)
- \( w_2 = 3 \)
- \( w_3 = 4 \)
- \( x_1 = 0 \)
- \( x_2 = 0 \)
- \( x_3 = 1 \)

- \( w_1 = 9 \)
- \( w_2 = 9 \)
- \( w_3 = 15 \)
- \( x_1 = 0 \)
- \( x_2 = 0 \)
- \( x_3 = 1 \)

- \( w_1 = 15 \)
- \( w_2 = 4 \)
- \( w_3 = 4 \)
- \( x_1 = 0 \)
- \( x_2 = 0 \)
- \( x_3 = 1 \)
Global Chance-Constraints

Filtering Algorithms for GCCs: An example

\[ \Pr(w_1 x_1 + w_2 x_2 + w_3 x_3 \leq 10) > 0.5 \]

Search Tree

Solution Tree

\[ x_1 = \{0, 1\} \]
\[ x_2 = \{0, 1\} \]
\[ x_3 = \{0, 1\} \]
Global Chance-Constraints

Filtering Algorithms for GCCs: An example

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

Search Tree

Solution Tree

- \( w_1 = 5 \)
- \( x_1 = \{0, 1\} \)
- \( x_2 = \{0, 1\} \)
- \( x_3 = \{0, 1\} \)

\( w_1 = 8 \)
\( w_2 = 3 \)
\( w_3 = 4 \)

\( w_2 = 9 \)
\( w_3 = 15 \)

\( x_1 = \{0, 1\} \)
\( x_2 = \{0, 1\} \)
\( x_3 = \{0\} \)
Global Chance-Constraints

Filtering Algorithms for GCCs: An example

\[ \Pr (w_1 x_1 + w_2 x_2 + w_3 x_3 \leq 10) > 0.5 \]

Search Tree

- \( x_1 = \{0,1\} \)
- \( x_2 = \{0,1\} \)
- \( x_3 = \{0,1\} \)

Solution Tree

- \( w_1 = 5 \)
- \( w_2 = 9 \)
- \( w_3 = 15 \)

\( x_1 = \{0,1\} \quad x_2 = \{0,1\} \quad x_3 = \{0\} \)

\( x_1 = \{0,1\} \quad x_2 = \{0,1\} \quad x_3 = \{0,1\} \)
Global Chance-Constraints

Filtering Algorithms for GCCs: An example

Pr(w₁x₁+w₂x₂+w₃x₃ ≤ 10) > 0.5

Search Tree

Solution Tree

x₁={0,1}  x₂={0,1}  x₃={0}

x₁={0,1}  x₂={0,1}  x₃={0,1}

x₁={0,1}  x₂={0,1}  x₃={0}

x₁={0,1}  x₂={0,1}  x₃={0,1}

x₁={0,1}  x₂={0,1}  x₃={0}

x₁={0,1}  x₂={0,1}  x₃={0,1}

x₁={0,1}  x₂={0,1}  x₃={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

\( x_1 = \{1\} \quad x_2 = \{0,1\} \quad x_3 = \{0\} \)
Filtering Algorithms for GCCs: An example

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

Search Tree

Solution Tree

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

$$w_1 = 5$$

$$w_2 = 3$$  $$w_3 = 4$$

$$w_1 = 8$$

$$w_2 = 3$$  $$w_3 = 4$$

$$w_2 = 9$$  $$w_3 = 15$$

$$w_3 = 15$$

$$w_1 = 15$$

$$w_2 = 9$$  $$w_3 = 15$$

$$w_2 = 3$$  $$w_3 = 4$$

$$w_3 = 4$$
Filtering Algorithms for GCCs: An example

\[ \Pr(w_1 x_1 + w_2 x_2 + w_3 x_3 \leq 10) > 0.5 \]
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

\[ \Pr(w_1 x_1 + w_2 x_2 + w_3 x_3 \leq 10) > 0.5 \]
Global Chance-Constraints

Filtering Algorithms for GCCs: An example

$$\Pr (w_1 x_1 + w_2 x_2 + w_3 x_3 \leq 10) \geq 0.5$$
Global Chance-Constraints

Filtering Algorithms for GCCs: An example

\[ \Pr (w_1x_1 + w_2x_2 + w_3x_3 \leq 10) \geq 0.5 \]
Filtering Algorithms for GCCs: An example

\[ \Pr(w_1 x_1 + w_2 x_2 + w_3 x_3 \leq 10) \geq 0.5 \]
Filtering Algorithms for GCCs: An example

\[ \Pr(w_1x_1 + w_2x_2 + w_3x_3 \leq 10) \geq 0.5 \]
Filtering Algorithms for GCCs: An example

Pr\( (w_1 x_1 + w_2 x_2 + w_3 x_3 \leq 10) \geq 0.5 \)
Filtering Algorithms for GCCs: An example

\[ P_r(w_1x_1 + w_2x_2 + w_3x_3 \leq 10) \geq 0.5 \]
Global Chance-Constraints

Filtering Algorithms for GCCs: An example

\[ \Pr(w_1x_1 + w_2x_2 + w_3x_3 \leq 10) \geq 0.5 \]
Global Chance-Constraints

Filtering Algorithms for GCCs: An example

\[ \Pr(w_1x_1 + w_2x_2 + w_3x_3 \leq 10) \geq 0.5 \]
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)
The model that uses GCC is much more compact!

<table>
<thead>
<tr>
<th>Stages</th>
<th>SBA Dec</th>
<th>SBA Cons</th>
<th>GCC Dec</th>
<th>GCC Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6484</td>
<td>6485</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>6554</td>
<td>6485</td>
<td>74</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>6739</td>
<td>6485</td>
<td>259</td>
<td>5</td>
</tr>
</tbody>
</table>
The test bed comprised, in total, 270 instances
To each instance we assigned a time limit of 240 seconds for running the search

<table>
<thead>
<tr>
<th>Stages</th>
<th>Solved Instances</th>
<th>Speed up</th>
<th>Node Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SBA</td>
<td>GCC</td>
<td>GCC</td>
</tr>
<tr>
<td>1</td>
<td>90</td>
<td>90</td>
<td>2.5×</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>45</td>
<td>13×</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>31</td>
<td>15×</td>
</tr>
</tbody>
</table>
Comparison

Run Times

![Run Times Graph](image-url)
Comparison

Explored Nodes
Comparison

Filtering

Domain Reduction

Percentage of decision variables assigned

Percentage of values pruned
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** → \( \Pr(w_1x_1 + w_2x_2 + w_3x_3 \leq 10) \geq 0.2 \)
Questions