Global Cost Functions in Weighted Constraint Satisfaction

[IJCAI'09, AAAI'10, ICTAI'11, JAIR'12, AAAI'12, ICTAI'12, CJ'14, AIJ'16]

Jimmy Lee, Mario Leung, Irwin Shum and Yi Wu

Department of Computer Science and Engineering

The Chinese University of Hong Kong

Shatin, N.T., Hong Kong

Global Cost Functions in Weighted Constraint Satisfaction

[IJCAI'09, AAAI'10, ICTAI'11, JAIR'12, AAAI'12, ICTAI'12, CJ'14, AIJ'16]

Jimmy Lee, Mario Leung, Irwin Shum and Yi Wu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

Global Cost Functions in Weighted Constraint Satisfaction

[IJCAI'09, AAAI'10, ICTAI'11, JAIR'12, AAAI'12, ICTAI'12, CJ'14, AIJ'16]

Jimmy Lee, Mario Leung, Irwin Shum and Yi Wu

Department of Computer Science and Engineering

The Chinese University of Hong Kong

Shatin, N.T., Hong Kong

Global Cost Functions in Weighted Constraint Satisfaction

[IJCAI'09, AAAI'10, ICTAI'11, JAIR'12, AAAI'12, ICTAI'12, CJ'14, AIJ'16]

Jimmy Lee, Mario Leung, Irwin Shum and Yi Wu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

Outline

1. Introduction
   - Motivation
   - Weighted constraint satisfaction
   - (Soft) global constraints

2. Global Cost Functions

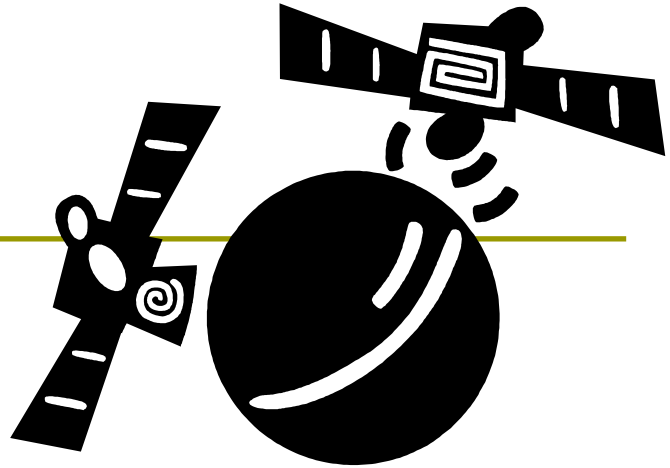
3. Towards a Library of Global Cost Functions

4. Concluding Remarks

Motivation

- Radio link frequency assignment
  - Given a **telecomunication network**
    - similar frequency for 2 nearby stations
      → interference!
    - Goal: assign the best frequency to each station to minimize interference

Motivation



- Satellite scheduling
  - **Spot 5** is an earth observation satellite with 3 on-board cameras
  - Request to take a set of pictures of different importance:
    - Resources, data-bus bandwidth, set-up times, orbiting, ...
  - Goal: select a subset of compatible pictures with maximum importance

Motivation

- DNA sequence alignment
  - Given ***k similar* DNA** (or protein) sequences, possibly with some noises
    - AAGAGGACAAGACCAGGGACTC
    - GAGGACAAGAAACCAAGGGTCAA
  - Goal: find the best alignment
    - AAGAGGACAAGA__CCA_GGGGACTC__
    - __GAGGACAAGAAACCAAGGG___TCAA

More Applications

- Combinatorial auctions
  - Given a set of goods and a set of bids. Find the best subset of bids to maximize the profits
- Crop allocation
  - Determine the crop growing sequence in a piece of farmland to satisfy crop rotation requirements, farmers' preferences and so on
- And many more …

Motivation

- Many real-life combinatorial problems involves costs
  - Restriction Violation → COST!
  - Preference → COST!
- We usually want a solution with the least cost
  - Least Violated
  - Most Preferred
- Soft Constraints or Cost Functions!
  - Assignments → COST

Motivation

- Handle soft constraints

  → Soft constraint frameworks

  - Fuzzy CSPs
  - Probabilistic CSPs
  - Weighted CSPs (WCSPs)
  - Semiring CSPs
  - …

Soft Constraints vs Cost Functions

- Given hard constraint $c(x_1,\ldots,x_n)$ and a violation measure.
- Soft constraint $c_{soft}(x_1,\ldots,x_n,z)$, where z represents the cost of tuple $(x_1,\ldots,x_n)$ [Petit, van Hoeve, Walsh, …]
- Cost function $\mathrm{c}_{cost}(x_1,\ldots,x_n)$ is directly the cost of tuple $(x_1,\ldots,x_n)$ [Schiex et al]

Weighted Constraint Satisfaction

- Variables :
  - $X = \{x_1, x_2, x_3\}$
- Domains:
  - $D(x_1) = D(x_2) = D(x_3) = \{a, b\}$
- Cost Functions:
  - Tuple → cost
  - $C_\varnothing = 0$
- Upper bound : k
  - $k = 4$

x_1	C_1
a	0
b	0

x_2	C_2
a	0
b	4

x_3	C_3
a	2
b	1

x_1	x_2	x_3	C_{123}
a	a	a	1
a	a	b	1
a	b	a	0
a	b	b	0
b	a	a	1
b	a	b	2
b	b	a	2
b	b	b	3

Cost Function Networks (CFNs)

- Variables :
  - $X = \{x_1, x_2, x_3\}$
- Domains:
  - $D(x_1) = D(x_2) = D(x_3) = \{a, b\}$
- Cost Functions:
  - Tuple $\rightarrow$ cost
  - $C_\emptyset = 0$
- Upper bound : k
  - $k = 4$

x_1	C_1
a	0
b	0

x_2	C_2
a	0
b	4

x_3	C_3
a	2
b	1

x_1	x_2	x_3	C_{123}
a	a	a	1
a	a	b	1
a	b	a	0
a	b	b	0
b	a	a	1
b	a	b	2
b	b	a	2
b	b	b	3

Cost Function Networks

$a \oplus b$
$= min(a+b, k)$

Cost of tuple $(a,a,a) = C_\varnothing \oplus C_1(a) \oplus C_2(a) \oplus C_3(a) \oplus C_{123}(a,a,a)$

Solution: cost $< k$ and minimum

$C_\varnothing = 0$ $k = 4$

x_1	C_1
a	0
b	0

x_2	C_2
a	0
b	4

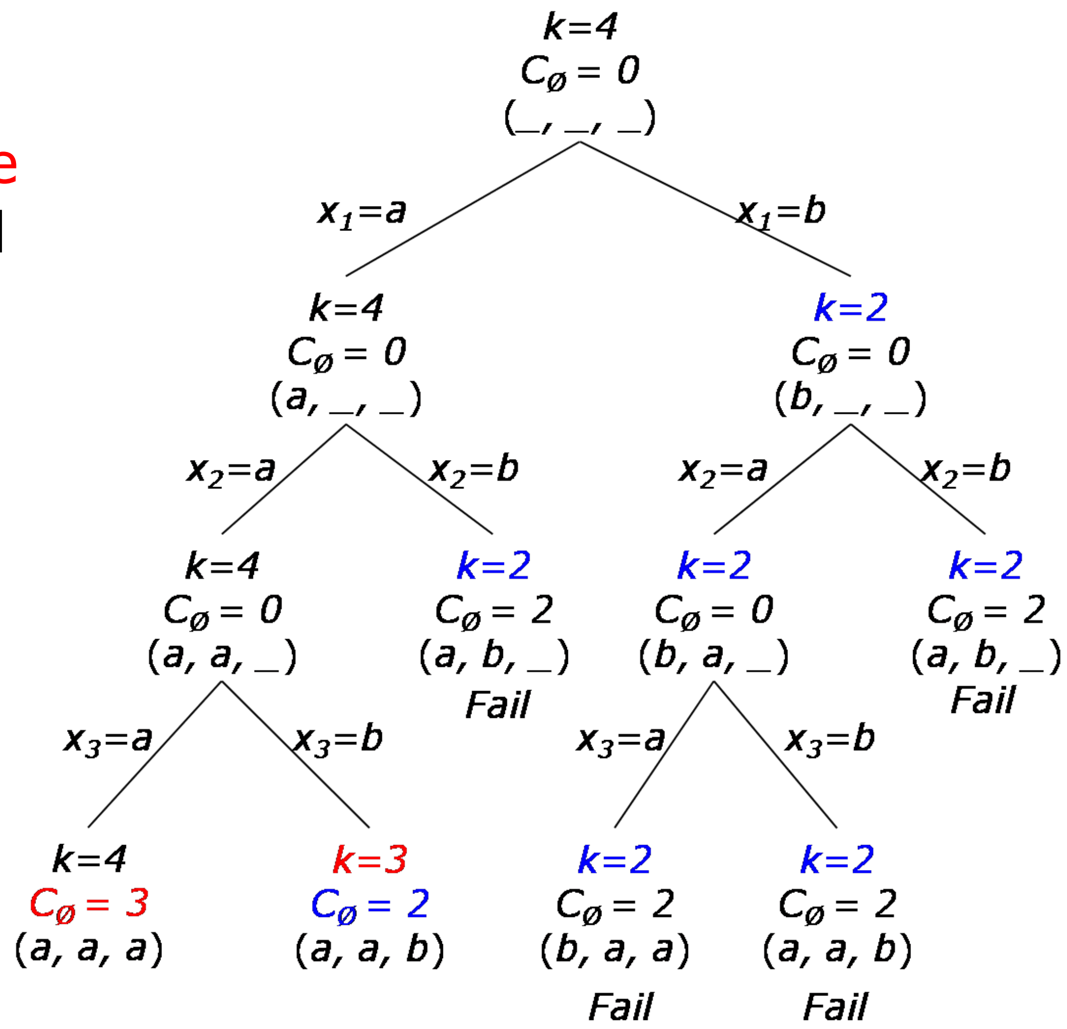
x_3	C_3
a	2
b	1

x_1	x_2	x_3	C_{123}
a	a	a	1
a	a	b	1
a	b	a	0
a	b	b	0
b	a	a	1
b	a	b	2
b	b	a	2
b	b	b	3

x_1	x_2	x_3	Cost of the tuple
a	a	a	3
a	a	b	2
a	b	a	4
a	b	b	4
b	a	a	3
b	a	b	3
b	b	a	4
b	b	b	4

Search for Solutions

- Solved by branch-and-bound searching
  - DFS with keeping the current best in k and searching for next best
- But search tree is too large



Consistency Algorithms

- Classical consistencies are notions of well-formedness (or cleaniness)

- Enforcement algorithms
  - Equivalence-preserving transformation
  - Extract implicit information and make them explicit, e.g. identifying values that cannot be part of any solution

- Prune search space and speed up search

Consistencies for CFNs

- Aim at redistributing costs among cost functions
  - Increase $C_{\emptyset}$
  - Remove infeasible values

- Better bounds and smaller search space

Cost Function Networks

Node Consistency (NC*) [Larrosa *et al.*, 02]

- $C_\varnothing \oplus C_i(v) < k$
- $\exists v \in D(x_i),\ C_i(v) = 0$

$k = 4$ $\quad \cancel{C_\varnothing = 0}$ $\quad C_\varnothing = 1$

project *1* from C_3 to $C_\varnothing$

x_2	C_2
a	*0*
~~*b*~~	~~*4*~~

NC*

x_3	C_3	
a	~~*2*~~	*1*
b	~~*1*~~	*0*

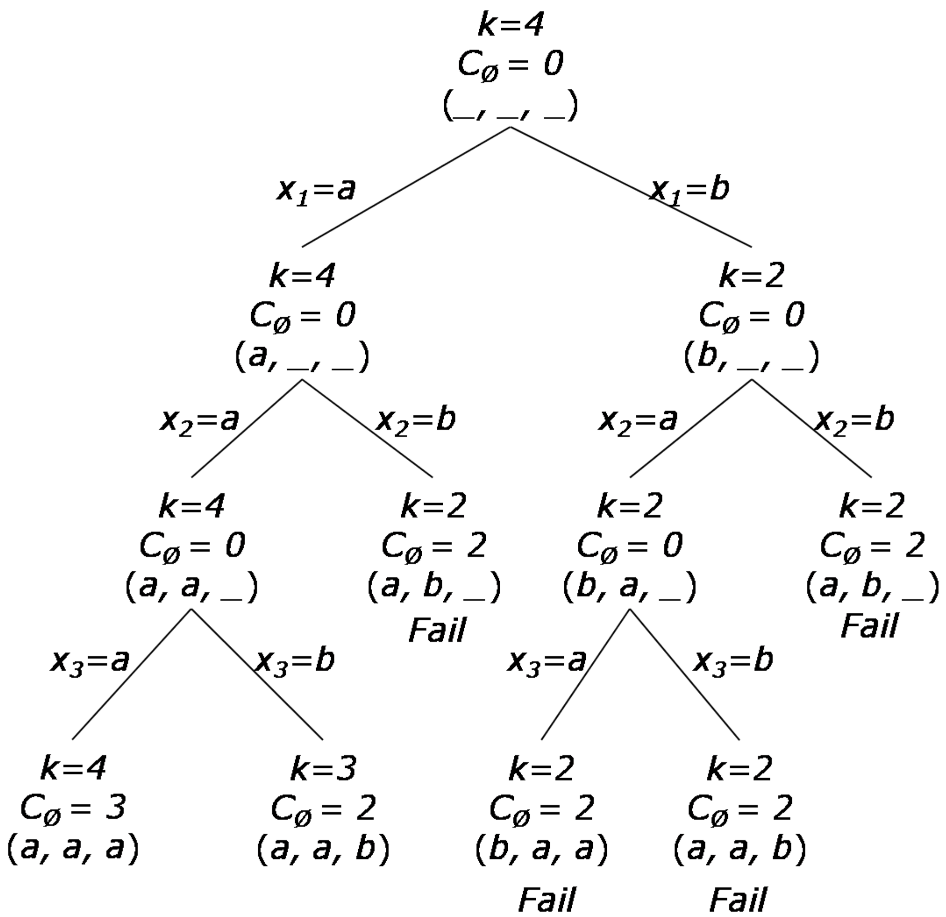
NC*

Cost Function Networks

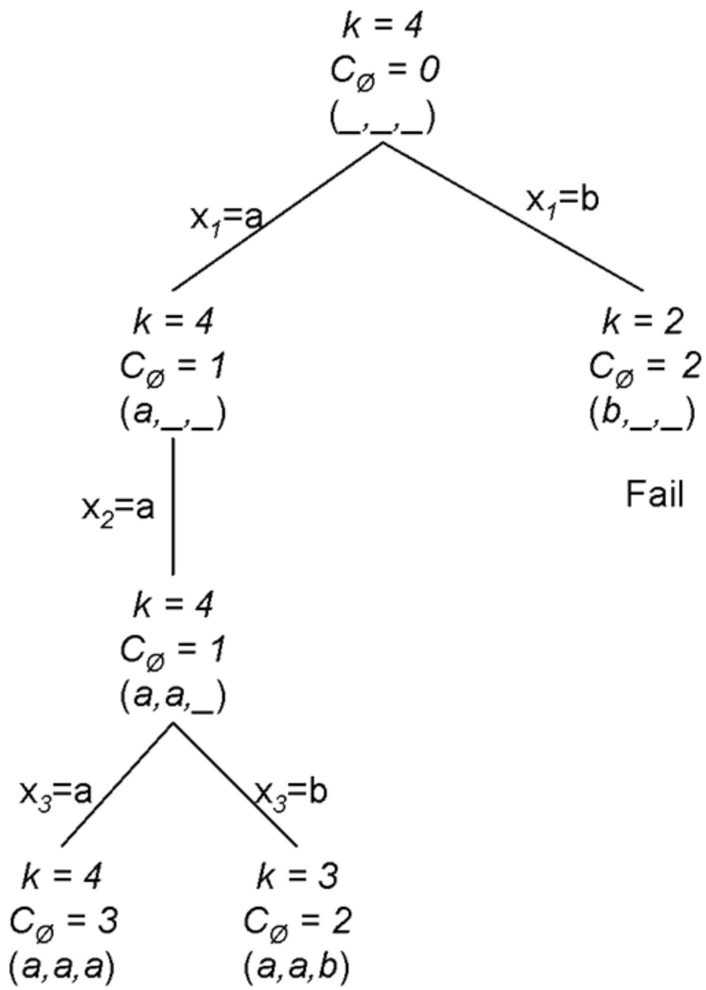
- AC*, FDAC*, EDAC* [Larrosa *et al.*, 02,03,04,05]
  - Deals with unary and binary constraints
  - Stronger than NC*
    - Detect more infeasible values
    - Give higher $C_\varnothing$

- By running consistency enforcement algorithms on each search node, the search space can be greatly reduced

Cost Function Networks



Without consistency



With consistency

Cost Function Networks

- AC*, FDAC* and EDAC* can be **generalized** for non-binary cost functions, but …
  - Solvers store constraints as tables
  - Efficient on binary and ternary constraints only
    - Time ↑ exponentially as # of variables **↑**

- **Can we bring the concept of global cost functions into CFNs?**

Global Constraints

- What are global constraints?
  - Constraints with special semantics capturing common sub-substructures in many problems

$$allDifferent([x_1, \ldots, x_n]) \leftrightarrow \bigwedge_{i \neq j} (x_i \neq x_j)$$

  - Varying number of arguments and usually of high arity
  - Usually with specialized polynomial time algorithms to enforce consistency in CSPs

Another Global Constraint Example

- Global Cardinality (GCC)
  - gcc([$x_1, \ldots, x_n$], [$v_1, \ldots, v_n$], [$c_1, \ldots, c_n$])
  - $c_i = \Sigma_{j=1..n} \ (x_j = v_i)$

- E.g. gcc(x, [1,2], [2,1])

  x = [1,1,2,3] ✅

  x =[1,2,3,4] ❌

- Global constraints are important for modeling real life problems. Some even argue that global constraints are the key to the success of CP!!

Global Cost Functions

- **Global Cost Functions**
  - Return violation cost
  - E.g. : *soft_allDifferent*^{dec} [Petit *et al.*, 01]
    - Return # of disequalities violated in $\{x_i \neq x_j \mid i > j\}$

allDifferent(a,c,c,c)

$a \neq c \wedge a \neq c \wedge a \neq c \wedge c \neq c \wedge c \neq c \wedge c \neq c$

soft_allDifferent$^{dec}(a,c,c,c)$ returns 3

Global Cost Functions

- Soft as Hard: Soft (global) constraints are used in constraint optimization [Petit et al. 00, Beldiceanu and Petit 04]
  - Extra weight variables
  - Aggregation of weight variables in objective
  - B&B with standard constraint propagation
- CFNs vs Soft as Hard
- How about incorporating **global cost functions** into CFNs?

Difficulties

- In current CFN solvers, cost functions are only represented as tables
- Consistency enforcement will have to check and modify an exponential number of tuples in case of high arity constraints
- **Does there exist a better representation rather than tables for global cost functions in CFNs?**

Outline

1. Introduction
2. Global Cost Functions in CFNs
   - GAC*
   - FDGAC*
   - Weak EDGAC*
   - Tractable Projection-safety
3. Towards a Library of Global Cost Functions
4. Concluding Remarks

Outline

1. Introduction
2. Global Cost Functions in CFNs
   - GAC*
   - FDGAC*
   - Weak EDGAC*
   - Tractable Projection-safety
3. Towards a Library of Global Cost Functions
4. Concluding Remarks

GAC*

GAC* [Copper and Schiex, 04]

- NC*
- $\forall\, v \in D(x_i)$ and a cost function C_S, $\exists$ tuple t with $t[x_i] = v$, $C_S(t) = 0$

$k = 4$ $\quad$ $C_{\varnothing} = 0$

x_1	x_2	x_3	C_{123}
a	a	a	1
a	a	b	0
a	b	a	0
a	b	b	1
b	a	a	$\cancel{1}$ 0
b	a	b	$\cancel{2}$ 1
b	b	a	$\cancel{2}$ 1
b	b	b	$\cancel{3}$ 2

x_1	C_1
a	0
b	$\cancel{0}$ 1

If $t[x_2] = b$,
$C_{123}(t) \geq 1$

project 1 from C_{123} to $C_1(b)$

GAC*

$k = 4$ $C_\varnothing = 0$

x_1	x_2	x_3	C_{123}
a	a	a	1
a	a	b	0
a	b	a	0
a	b	b	1
b	a	a	$\cancel{1}$ 0
b	a	b	$\cancel{2}$ 1
b	b	a	$\cancel{2}$ 1
b	b	b	$\cancel{3}$ 2

x_1	C_1
a	0
b	$\cancel{0}$ 1

If $t[x_2] = b$, $min\{C_{123}(t)\} = 1$

project 1 from C_{123} to $C_1(b)$

- Two steps:
  1) Compute $min\{C_S(t) | t[x_i] = v\}$
  2) Project from C_S to C_i

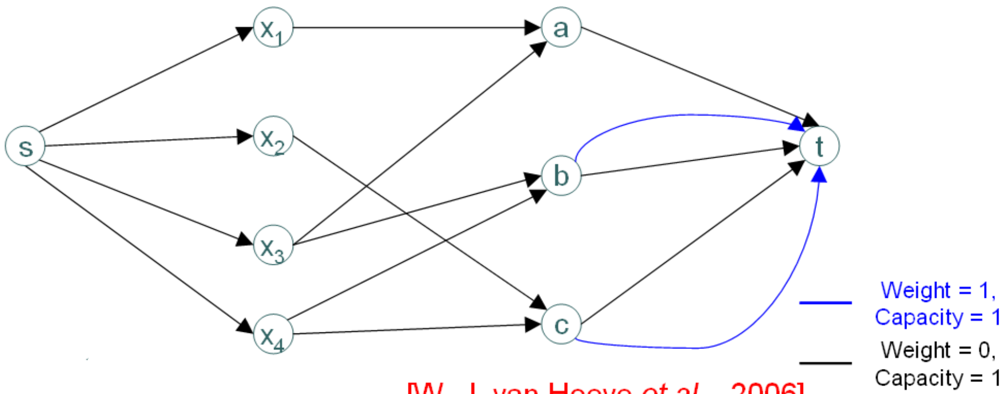
- We want:
  - Step *1*:
    - efficient
  - Step *2*:
    - efficient
    - maintain efficiency of step *1*

Many times!

GAC*

- Step *1*: flow-based [W.-J. van Hoeve *et al.*, 2006]
  - C_S ➔ minimum cost flow model
    - Minimum cost of the maximum flow = $min\{C_S(t)\}$
    - Polynomial time to compute $min\{C_S(t)\}$

soft_allDifferent^{dec} ➔



[W.-J. van Hoeve *et al.* , 2006]

GAC*

- Two steps:
  - Compute $min\{C_S(t)|t[x_i] = v\}$
    ➔ Efficient for flow-based constraints
  - Project from C_S to C_i
    ➔ ????

- Two problems for step 2

GAC*

1. Modify exponentially many tuples

x_1	x_2	x_3	C_{123}
a	a	a	1
a	a	b	0
a	b	a	0
a	b	b	1
b	a	a	~~1~~ 0
b	a	b	~~2~~ 1
b	b	a	~~2~~ 1
b	b	b	~~3~~ 2

$k = 4$ $C_{\varnothing} = 0$

x_1	C_1
a	0
b	~~0~~ 1

If $t[x_2] = b$,
$min\{C_{123}(t)\} = 1$

project 1 from C_{123} to $C_1(b)$

GAC*

2. The constraint is modified

x_1	x_2	x_3	C_{123}
a	a	a	1
a	a	b	0
a	b	a	0
a	b	b	1
b	a	a	1
b	a	b	2
b	b	a	2
b	b	b	3

projection →

x_1	x_2	x_3	C_{123}
a	a	a	1
a	a	b	0
a	b	a	0
a	b	b	1
b	a	a	0
b	a	b	1
b	b	a	1
b	b	b	2

Flow-based $\xrightarrow{\text{projection}}$ Flow-based?????

Tractable Projection Safety [IJCAI'09, JAIR'12]

- To deal with complexity problem
  ➔ Tractable projection-safety!

- A constraint C_S is tractable projection-safe:
  - C_S and C'_S are tractable, where C'_S is the new constraints formed by a series of valid projections (or extensions) from C_S

- Tractability is the basis for designing efficient and incremental consistency enforcing algorithm

Tractable Projection Safety by Flow

- A soft constraint is flow-based projection-safe if
  - **C_S is flow-based;**
  - **Each tuple in C_S maps to a unique flow**
  - **Each variable assignment maps to a unique set of edges in the flow network**

- A consequence is that projection can be done in constant time!

Flow-based Projection-Safety

- **Theorem**

  Assume C_S is flow-based projection-safe.

  After projecting from C_S to C_i,

  C_S is still flow-based projection-safe**.**

➔ Those constraints are **tractable projection-safe** w.r.t. the given three conditions.

- Flow-based projection-safety gives
  - sufficient conditions for tractable projection-safety
  - constant time projection

GAC*: Flow-based Projection-Safety

- **Theorem**

  If C_S is flow-based projection-safe, enforcing GAC* can be done in polynomial time in the size of the network

FDGAC* [IJCAI'09, JAIR'12]

- FDAC*[Larrosa *et al.*, 03] is stronger than AC*

- **Generalization** of FDAC* → FDGAC*

- **<u>Theorem</u>**

  FDGAC* is strictly stronger than GAC*

FDGAC*

- Enforcing FDGAC* requires extension
  - Reverse operation of projection
    - Projection: $C_S \rightarrow$ Unary constraints
    - Extension: Unary constraints $\rightarrow C_S$
  - Results for projection can be applied to extension!

FDGAC*

- **Theorem**

  If C_S is flow-based projection-safe, enforcing FDGAC* can be done in polynomial time in the size of the network

- But more expensive than that of GAC*…

- Can we have a consistency even stronger than FDGAC*?

Generalizing EDAC*

- EDAC* [de Givry *et al.*, 05] is stronger than FDAC*, but …

- Inherent limitation of EDAC*, especially prominent when adapted for non-binary constraints

  → may cause infinite looping!

Weak EDGAC* [AAAI'10, JAIR'12]

- To solve this problem, we **generalize** EDAC* to weak EDGAC* through cost-providing partitions

- **Theorem**

  Weak EDGAC* is strictly stronger than FDGAC*, GAC* for any cost-providing partitions

Weak EDGAC* [AAAI'10, JAIR'12]

- **Theorem 4**

  If C_S is flow-based projection-safe, enforcing weak EDGAC* can be done in polynomial time in the size of the network

- But it is more expensive to enforce than FDGAC*

Summary

- Different consistencies we have discussed:

Constraint Optimization

WCSPs

Increasing Strength

GAC

With flow-based projection-safe cost functions,
all can be enforced
efficiently in polynomial time

C

Weak EDGAC*

Outline

1. Introduction
2. Global Cost Functions in CFNs
3. Towards a Library of Global Cost Functions
   - C.f. Global Constraint Catalog [Beldiceanu and Carlsson, 03] with over 400 global constraints
4. Concluding Remarks

Towards a Library of Global Cost Functions

- Toulbar2 = Toulouse + Barcelona

- **Theorem**
  The following global cost functions are flow-based projection-safe:
  - **soft_allDifferent^{var} and soft_allDifferent^{dec}**
  - **soft_gcc^{var} and soft_gcc^{val}**
  - **soft_same^{var}**
  - **soft_regular^{var} and soft_regular^{edit}**

Beyond Flow-Based Projection-Safe

- Achieving tractable projection-safety other than using flow [AAAI'12, AIJ'16]

- How about min cost computation using LP? [ICTAI'11, ICTAI'12, CJ'14]

Polynomially DAG-Filterable [AAAI'12, AIJ'16]

- Theoretical characterizations of projections
- Polynomially DAG-Filterable Cost Functions are tractable projection-safe
- Efficient min cost computation using dynamic programming
- Soft variants of Among, Regular, Grammar, Max and Min

LP-based Consistencies [ICTAI'{11,12}, CJ'14]

- "NP-hard" global cost functions?
  - Difficult in nature: Soft variants of SlidingSum, eGCC, Disjunctive and Cumulative
  - Conjunctions of global cost functions: flow-based projection-safe cost functions

- Approximated forms of consistencies
- Polytime (Integral) Linear Projection-Safety: efficient approximated min cost computation using linear relaxation

Outline

1. Introduction
2. Global Cost Functions in CFNs
3. Towards a Library of Global Cost Functions
4. Concluding Remarks

Contributions

- A new algorithmic framework (tractable projection-safety) for designing and implementing global cost functions
- Four different consistencies
  1. Strong ∅IC (new!)
  2. GAC* (we make it practical!)
  3. FDGAC* (new!)
  4. Weak EDGAC* (new!)
- Extensive experimentations to verify and demonstrate efficiency and practicality

An Invitation

- Fill up the Global Cost Functions Catalog

- New data structures and algorithms
  - Flow graph and min cost flow
  - Polynomially sized DAG and dynamic programming
  - Matroids and greedy algorithms??

- New notions of tractable projection-safety

End

Q & A