Including Soft Global Constraints in Distributed Constraint Optimization Problems

Christian Bessiere
U. Montpellier
Montpellier, France

Patricia Gutierrez
Pedro Meseguer
IIIA-CSIC
Bellaterra, Spain

Overview

- Soft Global Constraints
- DCOP, solving algorithm: BnB-ADOPT+
- Representations of Soft Global Constraints
  - Direct / Nested / Bounded-arity
- Search with Soft Global Constraints
- Propagation with Soft Global Constraints
- Experimental Results
- Conclusions

Soft Global Constraints

- Soft Global =

  Global constraint *C* + violation measure μ

  tuple t, if t satisfies C, $\mu(t)=0$

  if t does not satisfy *C*, $\mu(t)>0$

- Example*:* *soft-alldifferent*(x_1,x_2,x_3,x_4)
  - μ_{var}: #variables should change value to satisfy the constraint
  - μ_{dec}: #pairs of variables with the same value

    soft-alldifferent(*a*,*a*,*a*,*b*), $\mu_{var}=2$, $\mu_{dec}=3$

DCOP

- (X, D, C, A, α)
- X is a set of variables
- D is a collection of finite domains
- C is a set of cost functions
- A is a finite set of agents
- α maps each variable to one agent (owner)
- Assumption: each variable to a different agent
- Solution: a total assignment of *minimum* cost

DCOP Optimal Solving Algorithms

- SBB, NCBB, DPOP, AFB,….
- BnB-ADOPT+:
  - agents in pseudo-tree
  - messages:
    - VALUE, from parent to child, pseudo-child
    - COST: from child to parent
    - TERMINATE: from parent to child
  - optimum: when *LB* = *UB* at root
- BnB-ADOPT+ combined with soft AC:
  - substantial performance improvements

Why Soft Global in DCOPs?

- Most DCOP works:
  - assume binary constraints
  - agents are usually constrained in pairs

- But
  - expressivity*:* *not every constraint* can be expressed as set of binaries, n-ary constraints are badly needed
  - efficiency: a soft global constraint
    - often prunes more that its decomposition
    - faster soft GAC (when solving includes soft GAC)

How including Soft Global in DCOPs?

- Direct: put the soft global constraint as it is
- Nested: for contractible soft global constraints
- Bounded-arity: for binary decomposable,
  or decomposable with extra variables
- Which offers the *best* performance?

Direct Representation

- Put the soft global constraint as it is
- Needed: a DCOP solving algorithm able to solve constraints of any arity
- BnB-ADOPT+: efficient solving n-ary constraints

Nested Representation

- For contractible soft global constraints:

$$C(x_1,\ldots,x_{k-1}) \leq C(x_1,\ldots,x_{k-1},x_k)$$

- Nested decomposition: $k-1$ constraints

$$C(x_1,x_2),\ C(x_1,x_2,x_3),\ \ldots\ldots,\ C(x_1,x_2,\ldots.x_k)$$

- Replace the soft global C by its nested decomp.

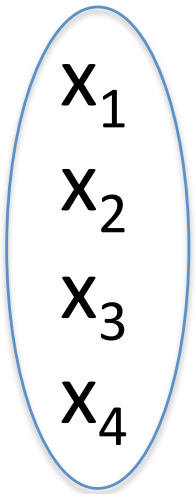
- Warning: not counting repeatedly the same costs

Bounded-arity Representation

- Decompositions in a polynomial number of constraints of fixed arity:
  - Binary decomposable without extra variables
  - Decomposable with extra variables
- Replace the soft global *C* by its decomposition

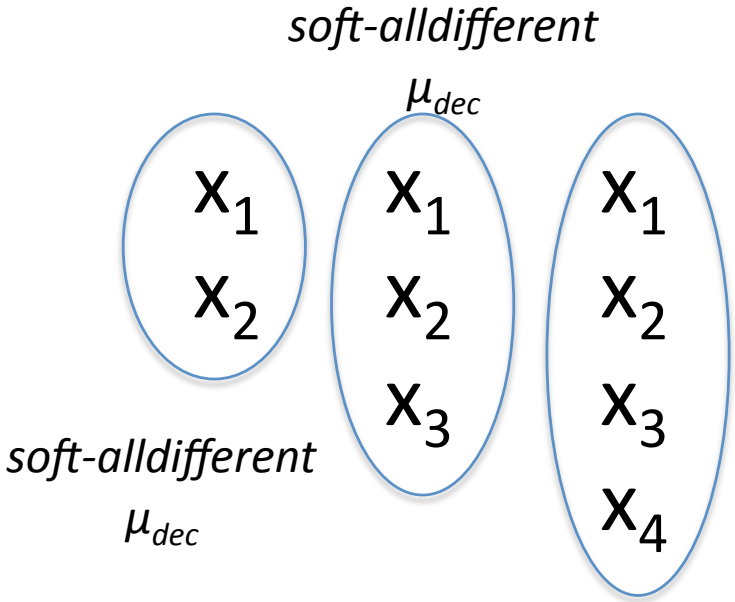
soft-alldifferent(x_1,x_2,x_3,x_4), μ_{dec}

direct



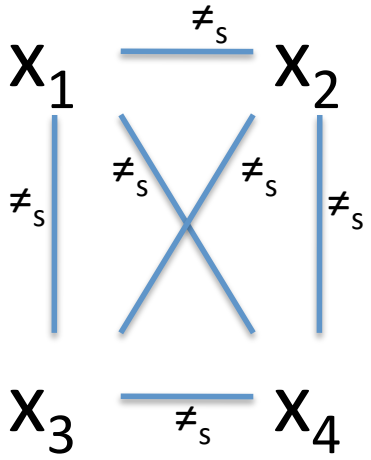
soft-alldifferent, μ_{dec}

nested



soft-alldifferent, μ_{dec}

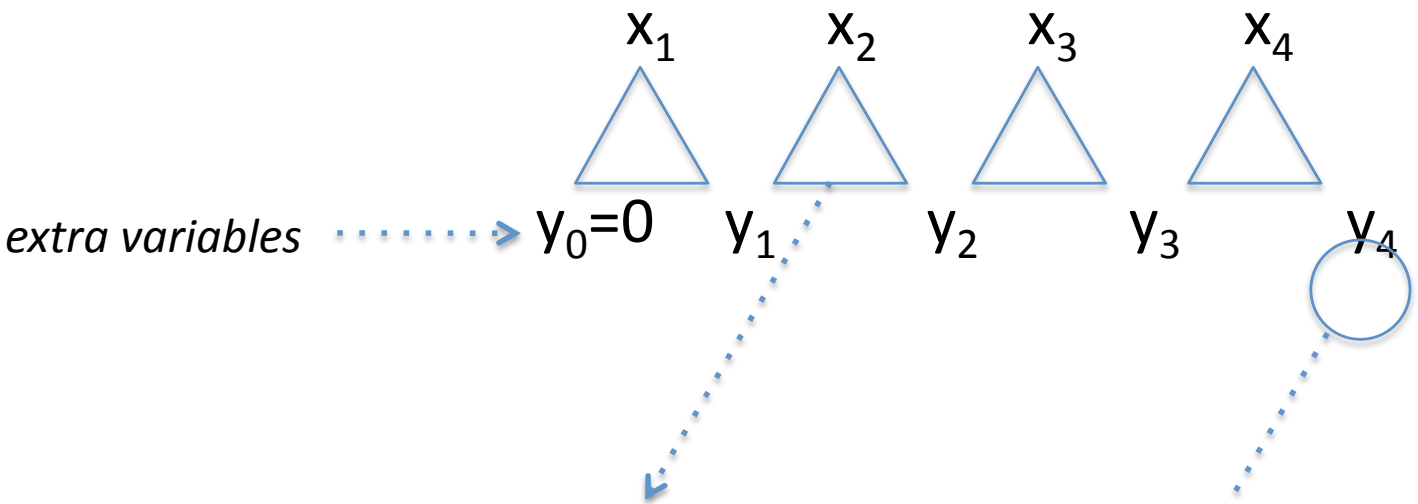
bounded arity



soft-atmost $[k,v](x_1,x_2,x_3,x_4)$, μ_{var}

If $\#v \leq k$ then $\mu_{var} = 0$ *else* $\mu_{var} = \#v - k$

Direct, nested: *as before* bounded-arity



Search with Soft Global Constraints

- BnB-ADOPT+: the last variable of the scope in the pseudo-tree branch evaluates
- Direct, bounded-arity: no problem
- Nested: simple trick to avoid counting twice the same costs

Soft Local Consistency

- Costs:
  - N-ary: $C_{ijk}(a,b,c)$
  - Unary $C_i(a)$
  - Zero-ary C_0:

soft local consistency

a lower bound of any solution cost

- NC: (x_i,a) *is NC* if $C_i(v) + C_0 < T$ (T = upper bound)
  x_i is NC if its values are NC and there is a st $C_i(a)=0$
  problem is NC if all variables are NC

- GAC: (x_i,a) is GAC wrt C_v if there is tuple t st (x_i,a) in t and $C_V(t)=0$
  x_i is GAC if all values are GAC wrt any cost function of x_i
  problem is GAC is every variable is GAC and NC

VALUE PRUNING: if a value is not NC it can be pruned

Propagating Soft Global Constraints

- Assumptions in DCOP solving: agent *i* knows
  - about its variable x_i
  - about the constraints it is involved in
  - nothing else
- Modifications for GAC in DCOP solving:
  - domain of neighbors: represented in agent *i* although agent *i* cannot delete a value in D_j
  - New DEL message to notify value deletions
  - BnB-ADOPT+: new info in VALUE & COST messages
- UGAC: unconditional deletions are propagated

Propagating *soft-alldifferent*(*T*)

- Flow-based global constraint [vanHoeve 06]

- Soft-alldifferent = flow graph
  - Minimum cost that can be projected = flow of minimum cost in the graph

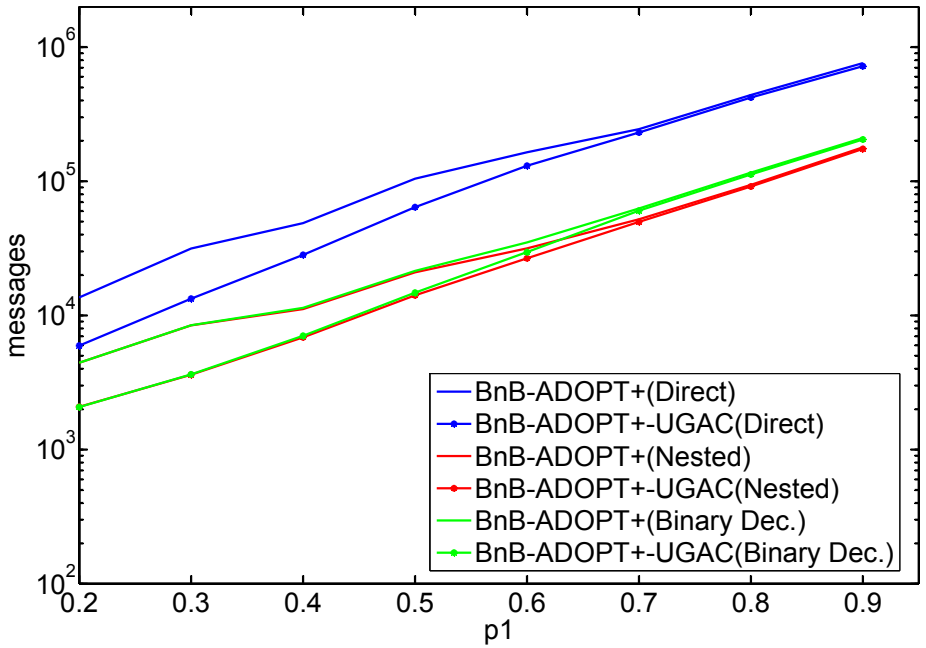
  - Projection-safe [Lee & Leung 09]

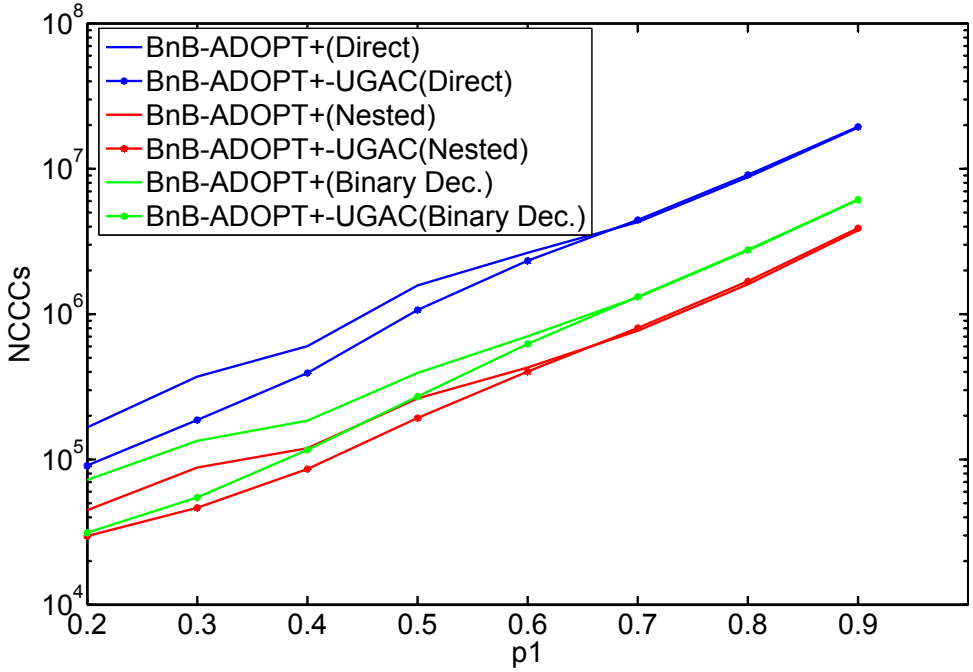
Propagating *soft-atmost* [*k*,*v*](*T*)

- Evaluator agent counts how many agents in *T* have singleton domains {*v*}
- If greater than *k*, a minimum cost #{*v*}-*k* is added to $C_i(a)$
- Agent *i*: first agent in the constraint

Experimental Results

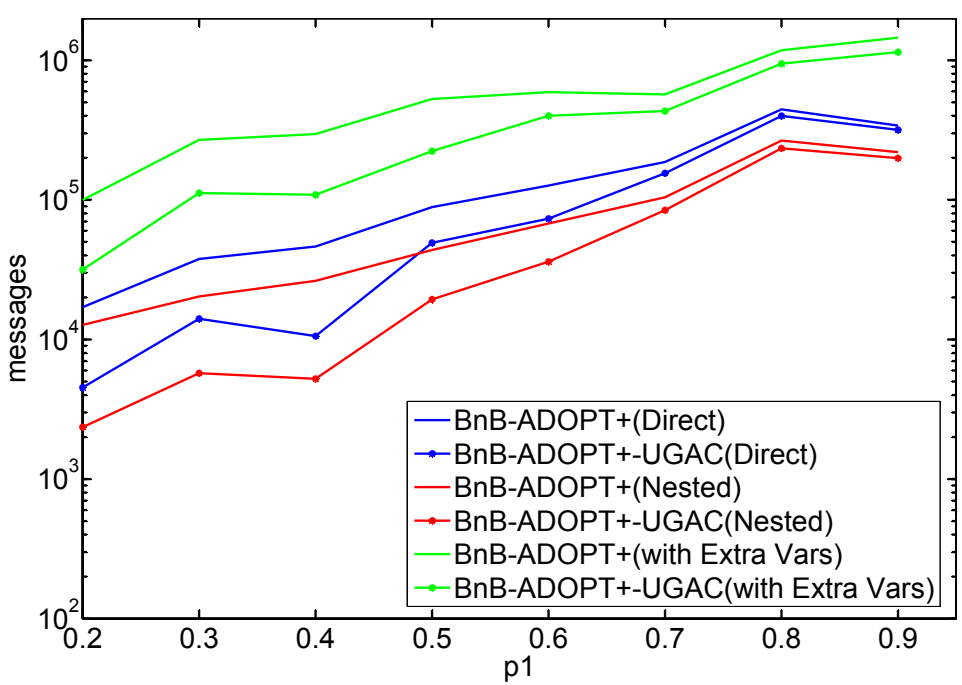
Benchmark: random binary <10,5,p_1> + 2 *soft-alldifferent*

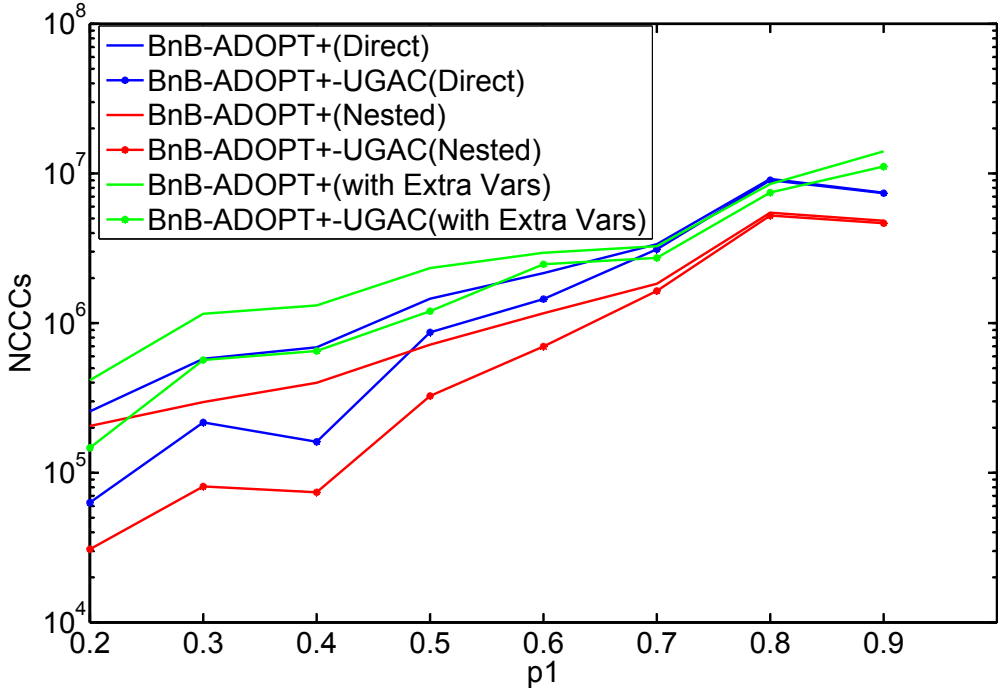




Experimental Results

Benchmark: random binary <10,5,p_1> + 2 *soft-atmost*





Conclusions

- Soft global constraints: needed in DCOP to increase *expressivity*

- With *soft-alldifferent* and *soft-atmost* as a proof of concept, we observe:
  - *Nested representation* is the most efficient (only for contractible soft global constraints)
  - UGAC *pays off* (always in #messages and in most cases in #NCCCs)



Thanks for your attention!