Title: On the Conversion between Non-Binary and Binary Constraint Satisfaction Problems
Authors: F. Bacchus and P. van Beek
Proc: AAAI 1998
Pages: 310–319

Foundations of Constraint Processing
CSCE421/821, Spring 2008
www.cse.unl.edu/~choueiry/S08-421-821/

Berthe Y. Choueiry (Shu-we-ri)
Avery Hall, Room 123B
choueiry@cse.unl.edu, Tel: (402)472-5444

Required reading:
On the Conversion between Non-Binary and Binary Constraint Satisfaction Problems, F. Bacchus and P. van Beek (AAAI’98)

Recommended reading: n-FC available from course URL

• On forward checking for non-binary constraint satisfaction.
  C. Bessière and P. Meseguer and E.C. Freuder and J. Larrosa,

• Decomposable Constraints.
  Ian Gent, Kostas Stergiou and Toby Walsh.
Summary

- Studies 2 mappings of non-binary CSPs into a binary representation \[ \{ \text{dual graph, hidden variable} \} \]
- Studies performance of BT search in each mapping vs. its performance in non-binary version
- Considers theoretical & experimental aspects
- Proposes FC\(^+\), yet lookahead strategy

Our goal: Learn about the mappings

Facts

- Non-binary constraints useful in the modeling of many applications
- Most research in CSPs is restricted to binary constraints
- Generalizing techniques for binary CSPs to address non-binary constraints is not straightforward
  - but sometimes done: FC & MAC
- Projection loses information
- Usual work-around/justification: (correctly) map non-binary constraints into binary ones
Ideally
- Modeling: use the most expressive/natural representation
- Solving: use the most 'effective' representation

PS: the 'effectiveness' of a representation per se is a new, and difficult, research area. No clear metrics exist, to my knowledge

Your options
- Directly apply techniques for non-binary CSP
  ...too few :-(
- Translate non-binary→binary, then solve
  Techniques for binary CSPs exploit graph/constraint properties
  Does the translation preserve/yield such properties?
  ...will the translation degrade the performance of the
techiques developed for binary CSPs?

Goal
- Study the effect of the translation on the performance of BT search
- Ultimately, establish properties of the translation to legitimize
  the restriction of research efforts to binary CSPs

Considers two translation methods

Results
- In most cases, the non-binary representation is most effective
- For tight constraints: binary representation wins
Example:

3SAT:
\[(X_1 \lor X_2 \lor X_6) \land (X_1 \lor X_3 \lor X_4) \land (X_4 \lor \bar{X}_5 \lor X_6) \land (X_2 \lor X_5 \lor \bar{X}_6)\]

3SAT as a non-binary (ternary) CSP

Variables: \(X_1, X_2, \ldots, X_6\)
Domains: \(D_{X_i} = \{0, 1\}\)
Constraints: \(C_{126} = \{(0, 0, 1), (0, 1, 0), \ldots\}\), except \((0, 0, 0)\)
\(C_{134} = \{0, 1\} \times \{0, 1\} \times \{0, 1\} \setminus \{(1, 0, 0)\}\)
\(C_{456} = \{0, 1\} \times \{0, 1\} \times \{0, 1\} \setminus \{(1, 1, 0)\}\)
\(C_{256} = \{0, 1\} \times \{0, 1\} \times \{0, 1\} \setminus \{(0, 0, 1)\}\)

FC for non-binary constraints

- A \(k\)-ary constraint is **forward-checkable**, if
  - \((k - 1)\) of its variables are instantiated
  - one variable uninstantiated
- BT-search:
  - instantiate one variable
  - repeat: for each newly f-checkable constraint, check future variable
  - if any domain is empty, backtrack
- Improvements: \(n\)-FC, \(n\)-FC2, \ldots, \(n\)-FC5
**Dual-graph representation**

**Usually:**
- CSP variable $\rightarrow$ node
- constraint $\rightarrow$ hyper-arc ‘label’

**Dual graph:**
- constraint $\rightarrow$ node (called c-variable)
- CSP variable $\rightarrow$ arc ‘label’

Constraint: $X_1$ must have the same value in $C_{126}$ and $C_{134}$
Domain of a c-variable: constraint definition

---

**Hidden-variable representation**

**Variables:**
- CSP variables +
  - 1 hidden variable (h-variable) per constraint

**Constraints:** only between a variable and the h-variables corresponding to its applicable constraints

Constraint: a value of $C_{126}$ correspond to one value of $X_1$
Domain of the h-variable = domain of the c-variable
Two binary representations

- **Dual graph**
  Nodes = only the constraints
  (CSP variables are not represented)
  Simple arcs between constraints

- **Hidden variable**
  Nodes = CSP variables and constraints
  Simple arcs constraints $\leftrightarrow$ variables

$\rightarrow$ Compare to Freuder’s constraint graphs

---

Theoretical comparison

I- Space requirements (data structures)
II- Analytical bounds (nodes, constraint checks in search)
I—Space requirements

- Binary representations require additional storing of domains for the c/h-variables (allowed k-tuples for each k-ary constraint)
  FC needs storage space proportional to the size of the domains (i.e., reductions)
  → could be substantial

- No space is needed to store constraints in binary representations: simple projection of an instantiation, can be done in constant time assuming domains of c/h-variables are stored extensionally

II—Analytical Bounds

Criteria
- number of visited nodes
- number of checks performed

Working assumption
- checking k-constraint costs k operations
- checking binary constraint costs 2 operations

Comparison
- dual-graph vs. non-binary
- hidden-variable vs. non-binary

Result
- not conclusive (one can always build a case where solving BT+FC has a better performance in one representation than in another)
- experimental evidence needed
Dual graph vs. non-binary CSP (I)

Loose constraint ⇒ exponentially large domains for c-variables ⇒ non-binary is less costly

Example:
n variables: $X_1, X_2, \ldots X_n$
n constraints: $X_1, \bar{X}_1 \lor X_2, \bar{X}_1 \lor X_2 \lor X_3, \ldots, \bar{X}_1 \lor \bar{X}_1 \lor \ldots X_n$

Non-binary: $n$ nodes, $\mathcal{O}(n^2)$ consistency checks
Dual-graph: $n$ nodes, $\mathcal{O}(2^n)$ consistency checks

Tight constraint ⇒ \ldots ⇒ dual-graph is less costly

Example:
n variables: $X_1, X_2, \ldots X_n$
n constraints: $X_1 \land \ldots \land X_{n-1}, X_1 \land \ldots \land X_{n-2} \land X_n, \ldots, X_2 \land \ldots \land X_n$

Non-binary: $2^{n-1}$ nodes, $\mathcal{O}(n2^n)$ consistency checks
Dual-graph: $n$ nodes, $\mathcal{O}(n^2)$ consistency checks

Improving FC: $FC^+$

- The constraint in the direction hidden-var→CSP-var is functional, but not vice-versa
- Search on hidden-var representation is restricted to the CSP-vars, h-vars used only for propagation
- FC is replaced with $FC^+$ to improve propagation
- $FC^+$ triggered improvements into nFC0, nFC1, \ldots, nFC5.
Experiments
Carried out on random CSPs
Results have predictive power verified by:
- random 3SAT
- crossword puzzles

Conclusions
Translating non-binary constraints involves overhead.
Translation is perhaps worthwhile if constraints are restrictive
Translation, as a strategy, is justifiable

Many open issues..
→ # tuples in constraints a good indicator? probably..
→ dual graph vs. hidden-variable?
→ .. we need to study further these translations/reformulations
→ to gain insight for designing good algorithms for
  non-binary constraints