Algorithm for Removing Redundant Edges in the Dual Graph of a Non-Binary CSP

Shant Karakashian
Constraint Systems Laboratory
University of Nebraska-Lincoln
Email: shantk@cse.unl.edu

Working Note 1-2008

November 10, 2008

1 Background

Below we describe the algorithm for finding the miminal dual graph of a non-binary CSP proposed in [1], because the pseudocode in the original paper is hard to parse.

2 Motivating Example

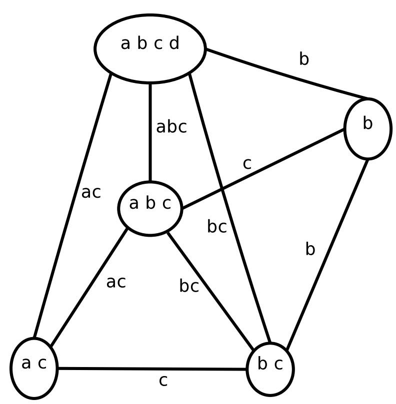

Figure 1: Dual graph of a CSP.

Consider the dual graph, $G = (C, E)$, of a non-binary CSP shown in Figure 1, where C is the set of non-binary constraints

$$C = \{c_i \mid c_i \text{ is a constraint}\} \tag{1}$$

and E is the set of edges between every two constraints with intersecting scopes.

$$E = \{(c_i, c_j) \mid Scope(c_i) \cap Scope(c_j) \neq \phi\} \tag{2}$$

In the above example, we have:

$$\begin{array}{rcl} C & = & \{c_{abcd}, c_{abc}, c_{ac}, c_{bc}, c_b\} \\ E & = & \{(c_{abcd}, c_{abc}), (c_{abcd}, c_{ac}), (c_{abcd}, c_{bc}), (c_{abcd}, c_b), \\ & & (c_{abc}, c_{ac}), (c_{abc}, c_{bc}), (c_{abc}, c_b), (c_{ac}, c_{bc}), (c_{bc}, c_b)\} \end{array}$$

The goal of the process is to find the minimal dual graph $G' = (E', C)$, where $E' \subseteq E$, and G' is equivalent to G and is obtained by removing as many redundancies from G as possible.
We assume that the original network is connected, othewise, we treat each connected component separately.

2.1 The set of overlaps

We denote O the set of 'overlaps' between the constraints as defined below:

$$O = \{o \,|\, o = Scope(c_i) \cap Scope(c_j), c_i, c_j \in C\} \tag{3}$$

In the example above, we have: O=$\{\{c\}, \{b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\}$.

2.2 Inducing a total ordering on the elements of O

We then sort the elements of O in decreasing size and denote them sorted elements as A_i where

$$A_i \subseteq A_j \Rightarrow |A_i| \leq |A_j| \Rightarrow j < i \tag{4}$$

In the example above, we have:

$$\begin{array}{rcl} A_1 & = & \{a, b, c\} \\ A_2 & = & \{a, c\} \\ A_3 & = & \{b, c\} \\ A_4 & = & \{b\} \\ A_5 & = & \{c\} \end{array}$$

Naturally, the ordering is not unique. Note that the ordering can be obtained by topological sorting using the partial order $\subseteq$ relation on the set O. However applying topological sorting would be more costly than simply sorting by the size of the elements of O.

2.3 Identifying the relations with identical overlaps

To each A_i, we associate the set α_{A_i}, which is the set of relations whose scope include the variables in A_i. α_{A_i} is defined as follows:

$$\alpha_{A_i} = \{c_i \mid A_i \subseteq Scope(c_i)\} \tag{5}$$

Note that the subgrapgh induced by α_{A_i} in the dual graph is necessarily complete.
In our running example, we have:

$$\begin{aligned}
\alpha_{A_1} &= \{c_{abcd}, c_{abc}\} \\
\alpha_{A_2} &= \{c_{abcd}, c_{abc}, c_{ac}\} \\
\alpha_{A_3} &= \{c_{abcd}, c_{abc}, c_{bc}\} \\
\alpha_{A_4} &= \{c_{abcd}, c_{abc}, c_{bc}, c_b\} \\
\alpha_{A_5} &= \{c_{abcd}, c_{abc}, c_{ac}, c_{bc}\}
\end{aligned}$$

2.4 Building the equivalent minimal graph

We now present the iterative process that builds the set E_{min} of the minimal graph. We start with the set $\gamma = \emptyset$, and grow the set γ_k at a step k until we reach E_{min}. We proceed as follows.

1. Given the dual graph $G = (C, E)$, extract the set O defined in Expression (3). The sets A_i and α_{A_i} are given in Expressions (4) and (5).

2. Let $\gamma \leftarrow \emptyset$

3. For $k \leftarrow 1$ to $|O|$, do

    - Construct the graph $G_k = (\alpha_{A_k}, \gamma)$
    - Construct the set $ConnectComp = \{\eta_i | \eta_i is\ a\ connected\ component\ of\ G_k\}$ of connected components of G_k
    - While $|ConnectComp| > 1$, do
        - Add any edge e_c that connects some $\eta_c, \eta_d \in ConnectComp$ to each other
        - $\gamma \leftarrow \gamma \cup \{e_c\}$
        - Replace η_c and η_d in $ConnectComp$ with the combined component
    - Return $ConnectComp$

Applying the above algorithm to the example in Figure 1 proceeds as follows. We set $\gamma \leftarrow \emptyset$ and O=$\{\{c\}, \{b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\}$.

1. - $G_1 = (\alpha_1, \lambda)$
    - $\alpha_{A_1} = \{c_{abcd}, c_{abc}\}$

- $\gamma = \emptyset$
- $ConnectComp = \{\{c_{abcd}\}, \{c_{abc}\}\}$
  - $\eta_c = \{c_{abcd}\}, \eta_d = \{c_{abc}\}, e_c = (c_{abcd}, c_{abc})$
  - $\gamma = \{(c_{abcd}, c_{abc})\}$
  - $ConnectComp = \{\{c_{abcd}, c_{abc}\}\}$

2. - $G_2 = (\alpha_2, \lambda)$
   - $\alpha_{A_2} = \{c_{abcd}, c_{abc}, c_{ac}\}$
   - $\gamma = \{(c_{abcd}, c_{abc})\}$
   - $ConnectComp = \{\{c_{abcd}, c_{abc}\}, \{C_{ac}\}\}$
     - $\eta_c = \{c_{abcd}, c_{abc}\}, \eta_d = \{c_{ac}\}, e_c = (c_{abcd}, c_{ac})$
     - $\gamma = \{(c_{abcd}, c_{abc}), (c_{abcd}, c_{ac})\}$
     - $ConnectComp = \{\{c_{abcd}, c_{abc}, c_{ac}\}\}$

3. - $G_3 = (\alpha_3, \lambda)$
   - $\alpha_{A_3} = \{c_{abcd}, c_{abc}, c_{bc}\}$
   - $\gamma = \{(c_{abcd}, c_{abc}), (c_{abcd}, c_{ac})\}$
   - $ConnectComp = \{\{c_{abcd}, c_{abc}\}, \{c_{bc}\}\}$
     - $\eta_c = \{c_{abcd}, c_{abc}\}, \eta_d = \{c_{bc}\}, e_c = (c_{abcd}, c_{bc})$
     - $\gamma = \{(c_{abcd}, c_{abc}), (c_{abcd}, c_{ac}), (c_{abcd}, c_{bc})\}$
     - $ConnectComp = \{\{c_{abcd}, c_{abc}, c_{bc}\}\}$

4. - $G_4 = (\alpha_4, \lambda)$
   - $\alpha_{A_4} = \{c_{abcd}, c_{abc}, c_{bc}, c_b\}$
   - $\gamma = \{(c_{abcd}, c_{abc}), (c_{abcd}, c_{ac}), (c_{abcd}, c_{bc})\}$
   - $ConnectComp = \{\{c_{abcd}, c_{abc}, c_{bc}\}, \{c_b\}\}$
     - $\eta_c = \{c_{abcd}, c_{abc}, c_{bc}\}, \eta_d = \{c_b\}, e_c = (c_{bc}, c_b)$
     - $\gamma = \{(c_{abcd}, c_{abc}), (c_{abcd}, c_{ac}), (c_{abcd}, c_{bc}), (c_{bc}, c_b)\}$
     - $ConnectComp = \{\{c_{abcd}, c_{abc}, c_{bc}, c_b\}\}$

5. - $G_5 = (\alpha_5, \lambda)$
   - $\alpha_{A_5} = \{c_{abcd}, c_{abc}, c_{ac}, c_{bc}\}$
   - $\gamma = \{(c_{abcd}, c_{abc}), (c_{abcd}, c_{ac}), (c_{abcd}, c_{bc}), (c_{bc}, c_b)\}$
   - $ConnectComp = \{\{c_{abcd}, c_{abc}, C_{ac}, c_{bc}\}\}\}$

$E_{min} \leftarrow \gamma$

References

[1] B. Nougier P. Jassen, Philippe Jégou and M.C. Vilarem. A filtering process for general constraint-satisfaction problems: Achieving pairwise-consistency using an associate binary representation. IEEE Workshop on Tools for Artificial Intelligence, pages 420–427, 1989.