Class Notes **Daniel Ruiz**
Prof. Wylie Date: 05/10/17

Subject: Constraint Logic Graphs, their types, and Logic.

As a game they are equivalent to a model of computation , in which they're used to represent the relations between constraints.

Definitions

A constraint graph is an oriented/directed graph with edge weights ϵ 1,2, which can be called red or blue, respectively. The inflow at each vertex is the sum of the weights on inward-directed edges. Each vertex has a non-negative minimum inflow. A legal configuration of a constraint graph has an inflow of at least the MI(minimum inflow) at each vertex.

Different Constraints, and Definitions:

Legal Moves:

In a constraint graph is the reversal of a single edge's orientation, which then results in another legal configuration.

Games on a Constraint Graph:

In a single player game, the goal becomes to reverse a given edge by executing a sequence of moves.

Multiplayer:

Each edge may only be controlled by one player, and each player may have a different goal.

Deterministic:

A unique sequence of reversals are forced upon the player.

Bounded:

An edge may only be reversed once, in which a constrain graph is an abstraction of a board game.

Gadgets

Gadgets are vertex configurations of AND/OR vertices, in which the minimum inflow of 2 incident edge weights are 1, 1, 2. Gadgets like Constraint Graphs are used to construct reductions from one computational problem to another, as part of proofs of

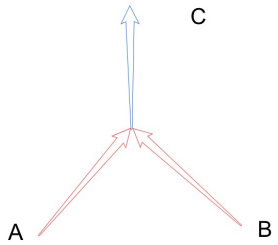
NP-completeness or other types of computational hardness such as P-Space, and P-Complete.

Types of vertices:

AND Vertex

An Edge C which is directed outwards if and only if edges A, B are pointed inwards.

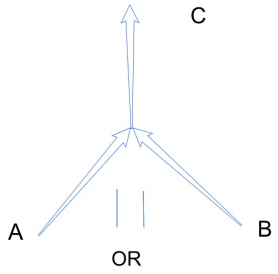
Figure 1: A picture of an AND vertex.



OR Vertex

An Edge C which is directed outwards if either edge A or B are pointed inwards.

Figure 2: A picture of an OR vertex.



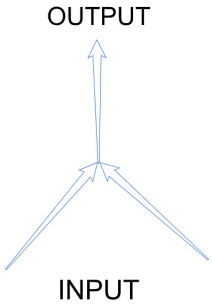
FANOUTS

A Fanout will be interpreted as a vertex with one ”input” and multiple ”outputs”.

AND/OR FANOUTS

Within a constraint graph it is thought as a digital logic circuit, which can then be thought as a model of computation.

Figure 3: A picture of a FANOUT.



Differences Between Gadgets and Constraint Graphs

In a digital Constraint Logic Graph, it is deterministic while the Constraint Logic has a degree of non-determinism, yet not on the same level. In a constraint logic we can wire the input/output to the output/input. In constraint logic there not an ¯not like vertex. Logic within the constraint graph is "monotone".

Definitions II:

Monotone:

A Boolean function is "monotone" if it contains literals, AND's, OR's. When a variable changes from false to true, the value of the formula can be change to TRUE to FALSE. Which is sufficient for computation.

Inflow:

Inflow then is a monotone function of incident edge orientation.

Vertex Sub-types:

Contains different versions of base vertices, which then make reductions easier to understand and create. For each kind of constraint logic we have a set of base vertices.

Planar Constraint Graph:

Crossover Gadgets:

Are used a lot in reductions, however these are some of the hardest to design due to their complexity.

Planars:

In a planar no edges cross, this allows for all constraint logic to have an equivalent constraint graph version. However, a Bounded deterministic constraint logic doesn't have a planar version. [1, p.19].

Figure 4: A picture of a full Crossover.

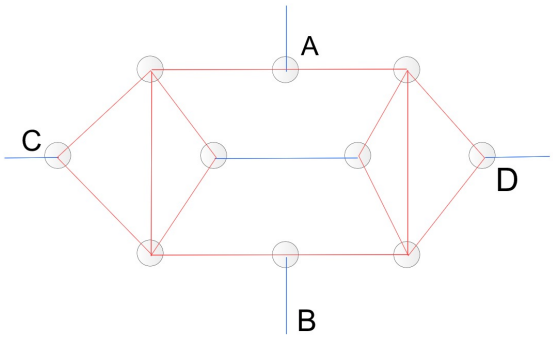


Figure 5: Here is a red-red-red-red or r^4, at least 2 points must point inwards, Only 2 edges point out, and the other two point inwards.

Figure 6: A picture of a Half-Crossover.

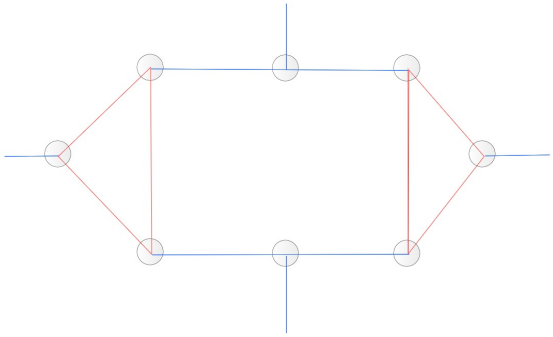


Figure 7: Equivalent to a r^4, can be plugged in exchange the Crossover to work just with AND/OR's

Choices, Conversions, and Edges:

Figure 8: A picture of a "Choice".

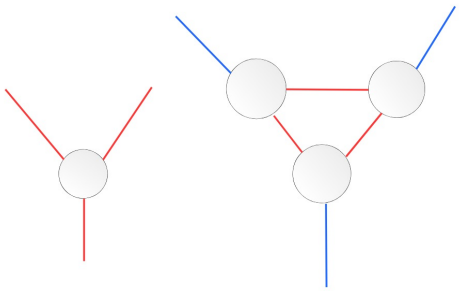


Figure 9: A picture of a Edge Conversions.

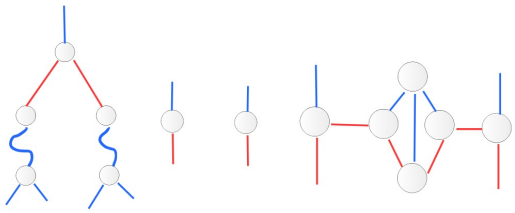
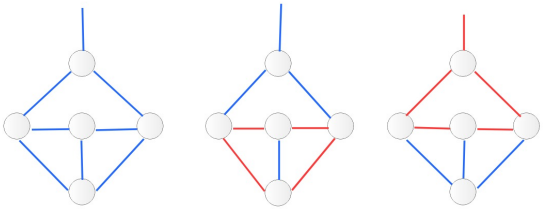


Figure 10: A picture of Free Edges.



Zero Player Game

Is a Bounded deterministic constant logic(not planar), in an instance in which a constant graph G_0, and the Edge R_0, and Edge R_e. A question rises, Is there an i such that if i is reversed on G_i?[1, p.43]

Deterministic Constraint Logic

Similar in many regards to a Bounded Constraint Logic, except that you look at all the ways to move on, if you can flip a node/path then you must do so ,and then move on. Thm. DCL is PSPACE complete for planar graphs.

One Player Games

Are Bounded non-deterministic Constraint Logic based. In the instance of a Constraint Graph G, with Edge E. Is there a sequence of moves on G, that eventually reverses E, such that each edge is reversed at least once?[1, p.44] Thm. NCL is PSPACE-Complete if it is planar. This is still true if it contains only protected OR's. Such that two of its edges due to global constraints cannot simultaneously be directed inwards.

Figure 11: A picture of a DCL.

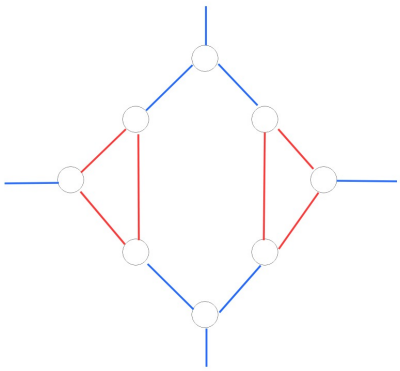
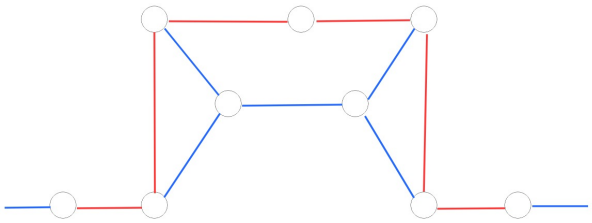


Figure 12: A picture of a Bounded NCL Edges.



Bounded two-player Constraint Logic

In the instance of a Constraint garph G, partition of the edges of G_0, into a partition of edges of G_i into sets B and W, and edges $e_\beta \epsilon B$, $e_\omega \epsilon \omega$. There arises the question of, does white have a forced win in the following game? Players Black and White alternate moves on G, they may only reverse moves on their own sets, the first to reverse $e_\omega(e_\beta)$ [1, p.91]

References

[1] Robert A. Hearn, Erik D. Demaine *Games, Puzzles, and Computations.*