CSCE 990 Advanced Constraint Processing
Scribe Notes of November 5, 2009
Speakers: Peter and Wesley
Scribe: Shant

This document summarizes the Path Consistency (PC) property and PC algorithms. It reviews triangulated graphs, and discusses PC algorithms on STPs. The material is summarized from Peter and Wesley's presentation.

1. Path Consistency as a consistency property

A CSP is PC iff it is strongly 3-consistent (van Beek & Dechter, JACM95). Given a PC CSP, every consistent solution over two variables can be extended to every third variable. The domains of the variables are filtered by AC.

In general, PC algorithms iterate over triplets of variables. In STP, variable domains are irrelevant, thus PC algorithms on STPs only enforce 3-consistency.

Typical PC algorithms determine if the CSP is path consistent, and may or may not filter the constraints as much as possible (e.g., DPC).

2. Properties of PC algorithms

Consider the PC algorithms: PC-1, PC-2, DPC, PPC, PC-8 and PC-2001. All PC algorithms stop when a relation or a domain is empty although this fact is not always specified in the pseudocode. PC algorithms may or may not have a queue. The queue, if used, may hold edges (e.g., PPC) or triplets of variables (PC-2) or tuples of 'vv-pair, variable' (e.g., PC-8 & PC-2001.)

The characteristics of a PC algorithm are described by

- Its ability to determine the PC property, i.e. whether or not the CSP is strong 3-consistent
- Its time and space complexities
- Its requirements for additional data structures for remembering supports
- The structure of the graph on which it operates (e.g., complete or chordal), and
- Its performance in practice.

Below I review each algorithm and its characteristics.

2.1 PC-1 (Mackworth 77)

The PC-1 algorithm has 4 nested loops that iterate over every combination of three variables until quiescence. It does not have a queue. It updates the edges and the domains by using composition and intersection. It determines strong 3-consistency in O(n^5d^5) time, and does not use additional data structures. It operates on complete graph.

2.2 PC-2 algorithm (Mackworth 77)

The PC-2 algorithm (Mackworth 77) uses a queue of triplets of variables, and iterates over the elements of the queue until the queue is empty. When an edge of a domain is updated, the triplets with the external third node are added to the queue. It determines strong 3-consistency. It is faster than PC-1 with time complexity O(n^3d^5), and space complexity O(n^3) for the queue, with no additional data structures. It operates on the complete graph.

2.3 PPC (Bliek & Sam-Haroud 99)

PPC uses a queue of edges. For each edge popped from the queue, it iterates over all triplets of variables related to that edge. If a relation corresponding to an edge in the triplet changes, then the edge is added to the queue. It operates on a chordal graph, hence it first triangulates the graph. PPC enforces strong PC, but has weaker filtering power than PC-2. The time complexity is O(δed^2) where δ is the degree of the graph. The space complexity is O(δe). If two or more edges of a triplet are in the queue, all three edges are redundantly updated once for each edge in the queue. The redundant work is avoided in △STP.

2.4 DPC (Dechter & Pearl 89)

DPC operates on a given ordering of variables. It traverses the variables bottom to top in the given ordering first enforcing DAC. On a second pass, it update the edge between every two of the parents of the traversed variable. As a result, it moralizes the graph and determines strong directional path consistency relative to the ordering. The time complexity is O(min($t.d^3$,n^3d^3)), and does not use any additional data structures. It operates on chordal graph.

2.5 PC-8 (Chmeiss & Jegou 98)

PC-8 uses a queue of (vvp, variable) elements, and determines strong PC achieving full filtering. The time complexity is O(n^3d^4) and the space complexity is O(n^2d). It operates on a complete graph. PC-8 outperforms PC-2 at phase transition.

2.6 PC-2001 (Bessiere+ 05)

PC-2001 uses the same queue as in PC-8, but in addition records supporting values to improve time complexity. Hence it achieves the same properties as in PC-1, PC-2 and PC8 in time complexity of O(n^3d^3) and space complexity O(n^3d^2). It operates on complete graph. PC-2001 is faster than PC-8, but has worse space complexity.

3. Triangulated Graphs

The speakers

The speakers recalled various definitions relative to triangulated graphs:

- A triangulated graph is a graph with a chord for every cycle of length four or more.
- Every triangulated graph has a *perfect elimination ordering* (PEO), and every graph that has a PEO is triangulated.
- PEO is an ordering of the graph vertices that when removed one vertex at a time in that order, each removed vertex is a *simplicial* vertex at the time of removal.
- A vertex is simplicial if all of its neighbors are connected.
- Fill in edges are the edges that need to be added to make a vertex simplicial.
- The width of a triangulated graph is equal to the size of the its largest clique-1.
- The induced width of the PEO is the width of the triangulated graph.
- The reverse of the PEO yields a moralized graph.

Triangulated graphs are relevant for the study of PC algorithms for the following reasons:

- DPC operates on a moralized graph (in fact it moralizes the graph).
- PPC by Bliek & Sam-Haroud operates on triangulated graphs, determines the property of strong PC, and when the constraints are convex, yields minimal CSP.

4. PC on STPs

Floyd-Warshall, Bellman-Ford, △STP and P^{3}C are algorithms for enforcing PC on STPs.

- Floyd-Warshall is a basic STP solver with three nested loops. If no edge is found between a pair of variables, adds and edge with infinite distance. The time complexity is $\Theta(n^3)$. Operates on complete graph.
- Bellman-Ford does not guarantee minimality, but detects inconsistency. The time complexity is O(en).
- △STP by Xu & Choueiry adapts PPC to CPSs without updating the domains. △STP maintains a queue of triangles instead of queue of edges. Similarly to

PPC, it operates on triangulated graphs, and fully filters the constraints because they are convex. It keeps a queue of triangles, and for each popped triangle updates all the three edges. When an edge is updated, it enqueues only the triangles adjacent to the updated edge. Performance is best when the queue is FIFO. Experiment results show that △STP performs better than PPC, and PPC performs better than F-W. Planken et al. showed that △STP is $\Omega(t^2)$ on a pathological case, t is the number of triangles, and proposed P^{3}C, that is O(t). A more accurate bound for △STP is O(min(t^2, $\delta e d^2$)).

5. P³C

Designers of △STP became aware of the relevance of perfect elimination ordering in △STP in 2005 (ref. Nic Wilson), and the designers of Prop-STP exploited this idea. The authors of P^{3}C formalized the flaw of △STP, and proposed a two phase algorithm over a given perfect elimination ordering to achieve the filtering equivalent to △STP.

1. In the first phase, it applies DPC bottom up, considering every pair of parents for each variable in the ordering, and updates the edge between the parents.
2. In the second phase, moves top down, considering every pair of parents for each node and updates the edges adjacent to the node.

In each phase, P^{3}C visits each triangle once, hence it runs in Θ(t) time.

6. Discussions

- △STP: when a triangle that is to be added because of an update is already in the queue, it is not added, and the triangle in the queue keeps its position.
- △STP operates within each biconnected component (“does not pass through articulation points”) and the domains are not updated.
- △STP and P^{3}C have the same time when applied on a graph with enforced consistency. This is because both algorithms visit each triangle at most twice when there are no updates. △STP may re-visit triangles in case of updates, while P^{3}C does not.
- The proof of correctness of the P^{3}C algorithm was extensively discussed during the presentations.