Progress on Certifying Algorithms

Kurt Mehlhorn and Pascal Schweitzer

Max-Planck-Institut für Informatik, Saarbrücken, Germany

A *certifying algorithm* is an algorithm that produces with each output, a *certificate* or witness (easy-to-verify proof) that the particular output has not been compromised by a bug. A user of a certifying program P (= the implementation of a certifying algorithm) inputs x, receives an output y and a certificate w, and then checks, either manually or by use of a checking program, that w proves that y is a correct output for input x. In this way, he/she can be sure of the correctness of the output without having to trust P. We refer the reader to the recent survey paper [9] for a detailed discussion of certifying algorithms.

1 An Example

We illustrate the concept by an example. A *matching* in a graph G is a subset M of the edges of G such that no two share an endpoint. A matching has maximum cardinality if its cardinality is at least as large as that of any other matching. Figure 1 shows a graph and a maximum cardinality matching. Observe that the matching leaves two nodes unmatched, which gives rise to the question whether there exists a matching of larger cardinality. What is a witness for a matching being of maximum cardinality? Edmonds in his seminal papers [1,2] on how to compute maximum matchings in polynomial time introduced the following certificate: An *odd-set cover OSC* of G is a labeling of the nodes of G with nonnegative integers such that every edge of G is either incident to a node labeled 1 or connects two nodes labeled with the same number $i \geq 2$.

Theorem 1 ([1]). *Let N be any matching in G and let OSC be an odd-set cover of G. For any $i \geq 0$, let n_i be the number of nodes labeled i. Then*

$$|N| \leq n_1 + \sum_{i \geq 2} \lfloor n_i/2 \rfloor.$$

Proof. For i, $i \geq 2$, let N_i be the edges in N that connect two nodes labeled i and let N_1 be the remaining edges in N. Then

$$|N_i| \leq \lfloor n_i/2 \rfloor \text{ for } i \geq 2 \quad \text{and} \quad |N_1| \leq n_1$$

and the bound follows.

It can be shown (but this is non-trivial) that for any maximum cardinality matching M there is an odd-set cover OSC with

$$|M| = n_1 + \sum_{i \geq 2} \lfloor n_i/2 \rfloor, \tag{1}$$

thus certifying the optimality of M. In such a cover all n_i with $i \geq 2$ are odd, hence the name.

A *certifying algorithm for maximum cardinality matching* returns a matching M and an odd-set cover OSC such that (1) holds. Edmonds [1,2] gave the first efficient algorithm for maximum cardinality matchings. The algorithm is certifying and outputs a matching M and an odd-set cover OSC satisfying (1). By the argument above, the odd-set cover proves the optimality of the matching. Observe, that is it *not* necessary to understand why odd-set covers proving optimality always exist. It is only required to understand the simple proof of Theorem 1, showing that equation (1) proves optimality. Also, a correct checking program which controls whether a set of edges is a matching and a node labelling is an odd-set cover which together satisfy (1) is easy to write.

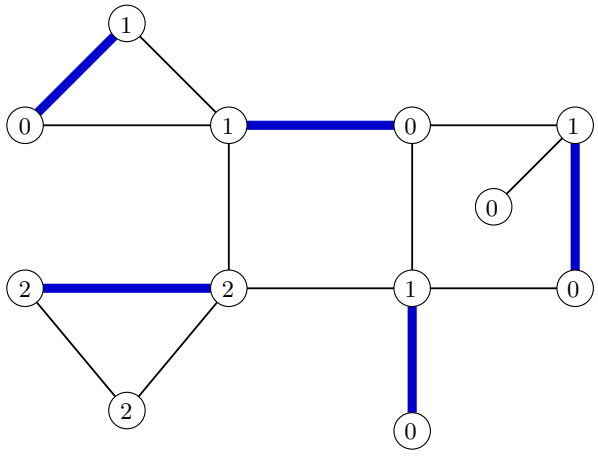

Fig. 1. The node labels certify that the indicated matching is of maximum cardinality: All edges of the graph have either both endpoints labelled as two or at least one endpoint labelled as one. Therefore, any matching can use at most one edge with two endpoints labelled two and at most four edges that have an endpoint labelled one. Therefore, no matching has more than five edges. The matching shown consists of five edges.

2 Formal Verification

We repeat the last two sentences of the introduction. It is only required to understand the simple proof of Theorem 1, showing that equation (1) proves optimality. Also, a correct program which checks whether a set M of edges is a matching and a node labelling OSC is an odd-set cover which together satisfy (1) is easy to write. Such a program would have to verify that M is a subset of E, that the degree of every vertex with respect to M is at most one, that the node labelling OSC is an odd-set cover, and that (1) holds.

If the preceding paragraph holds true, we should be able to substantiate it in two ways:

1. Turn the proof of Theorem 1 into a formal proof.
2. Prove the correctness of the checking program.

In ongoing work of the first author with Eyad Alkassar, Christine Rizkallah, and Norbert Schirmer we have done exactly this. We formalized and proved Theorem 1 using Isabelle [7] and we wrote the checker in Isabelle and C and proved it correct using Isabelle and VCC [14], respectively. We plan to extend this work to a large fraction of the certifying algorithms covered in [9].

3 Three-Connectivity of Graphs

An undirected graph is 3-connected if the removal of any two vertices does not disconnect it. Linear time algorithms for this problem are known [6,10], but none of them are certifying. The fastest certifying algorithms have quadratic running time $O(n^2)$, see [12,9]. We review the algorithm given in the latter paper.

Certifying that a graph is not 3-connected is simple; it suffices to provide a set S of vertices, $|S| \leq 2$, such that $G \setminus S$ is not connected. The non-certifying algorithms [6,10] above compute such a set if the input is not 3-connected. However, if the input is 3-connected, the only output returned is "input is 3-connected". Gutwenger et. al. [5] implemented the algorithm of [6] and report that it incorrectly declares some non-3-connected graphs 3-connected. They provide a correction.

Tutte [13] introduced a certificate for 3-connectedness, a sequence of edge contractions resulting in the K_4, the complete graph on 4 vertices. We call an edge e of a 3-connected graph G *contractible*

if the contracted graph G/e, (i.e. the graph obtained by replacing the end-vertices of e by a single vertex neighboring all vertices previously adjacent to one of the endpoints) is 3-connected. A separating pair is a pair of vertices whose removal disconnects the graph.

Lemma 1. *Let $e = (x, y)$ be an edge of a simple graph G whose end-vertices have a degree of at least 3. If G/e is 3-connected, then G is 3-connected.*

Proof. Since contractions cannot connect a disconnected graph, the original graph G is connected. There are no cut-vertices in G, as they would map to cut-vertices in G/e.

Any separating pair of G must contain one of the end-vertices of edge e. Otherwise the pair is also separating in G/e. It cannot contain both x and y, otherwise the contracted vertex xy is a cut-vertex in G/e. Suffices now to show that x, u, with $u \in V(G) \setminus y$ is not a separating pair. Suppose otherwise, then the graph $G - \{x, u\}$ is disconnected, but the graph $G\{x, y, u\}$ is not. Thus $\{y\}$ is a component of $G - \{x, u\}$. But this is a contradiction since y has degree at least 3 in G.

To certify the 3-connectivity of a graph G, it thus suffices to provide a sequence of edges which, when contracted in that order, have endpoints with a degree of a least 3 and whose contraction results in a K_4. We call such a sequence a *Tutte sequence*. We now focus on how to find the contraction sequence, given a 3-connected graph.

The $O(n^2)$ algorithm needs three ingredients: First we require the $O(n^2)$ algorithm by Nagamochi and Ibaraki [11] that finds a sparse spanning 3-connected subgraph of G with at most $3n - 6$ edges. Second we require a linear time algorithm for 2-connectivity. Third we require a structure theorem, that shows how to determine a small candidate set of edges among which we find a contractible edge.

Theorem 2 (Krisell [8]). *If no edge incident to a vertex v of a 3-connected graph G is contractible, then v has a least four neighbors of degree 3, which each are incident with two contractible edges.*

Consider now a vertex v of minimal degree in a 3-connected graph. If it has degree three, it cannot have four neighbors of degree three and hence must have an incident contractible edge. If it has degree four or more, it cannot have a neighbor of degree three (because otherwise, its degree would not be minimal) and hence must have an incident contractible edge. Also note that an edge xy in a 3-connected graph is contractible, if $G - \{x, y\}$ is 2-connected.

We explain how to find the first $n/2$ contractions in time $O(n^2)$. By repeating the procedure we obtain an algorithm that has overall a running time of $O(n^2)$.

First use the algorithm by Nagamochi and Ibaraki [11]. The resulting graph has $3n - 6$ edges. Thus while performing the first $n/2$ contractions, there will always be a vertex with degree at most $2 \cdot 2 \cdot 3 = 12$. Choosing a vertex of minimal degree, we obtain a set of at most 12 candidate edges, one of which must be contractible. To test whether an edge xy is contractible, we check whether $G - \{x, y\}$ is 2-connected with some linear time algorithm for 2-connectivity.

Theorem 3 ([12]). *A Tutte sequence for a 3-connected graph can be found in time $O(n^2)$.*

It remains a challenge to find a linear time certifying algorithm for 3-connectivity of graphs. A linear time certifying algorithm for graphs that contain a Hamiltonian cycle was recently found [3]; this assumes that the Hamiltonian cycle is part of the input. It was also shown recently [4] that for any DFS-tree T of a 3-connected graph G, there is a Tutte sequence that contracts only edges in T.

References

1. J. Edmonds. Maximum matching and a polyhedron with 0,1 - vertices. *Journal of Research of the National Bureau of Standards*, 69B:125–130, 1965.

2. J. Edmonds. Paths, trees, and flowers. *Canadian Journal on Mathematics*, pages 449–467, 1965.
3. A. Elmasry, K. Mehlhorn, and J. M. Schmidt. A Linear Time Certifying Triconnectivity Algorithm for Hamiltonian Graphs. Available at the authors home pages, March 2010.
4. A. Elmasry, K. Mehlhorn, and J. M. Schmidt. Every DFS-Tree of a 3-Connected Graph Contains a Contractible Edge. Available at the authors home pages, Februar 2010.
5. C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. In *Graph Drawing*, LNCS, pages 77–90, 2000.
6. J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal of Computing*, 2(3):135–158, 1973.
7. Isabelle theorem prover. `http://isabelle.in.tum.de/`.
8. M. Kriesell. A survey on contractible edges in graphs of a prescribed vertex connectivity. *Graphs and Combinatorics*, pages 1–33, 2002.
9. K. Mehlhorn, R. McConnell, S. Näher, and P. Schweitzer. Certifying Algorithms. available at the first author's homepage and submitted for publication.
10. G. L. Miller and V. Ramachandran. A new graph triconnectivity algorithm and its parallelization. *Combinatorica*, 12(1):53–76, 1992.
11. H. Nagamochi and T. Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7:583–596, 1992.
12. J. M. Schmidt. Construction sequences and certifying 3-connectedness. In *27th International Symposium on Theoretical Aspects of Computer Science (STACS'10)*, Nancy, France, 2010. `http://page.mi.fu-berlin.de/jeschmid/pub`.
13. W. Tutte. A theory of 3-connected graphs. *Indag. Math.*, 23:441–455, 1961.
14. VCC, a mechanical verifier for concurrent C programs. `http://vcc.codeplex.com/`.