A Faster Test for 4-Flow-Criticality in Snarks

André Breda Carneiro [a] Cândida Nunes da Silva [a]
Brendan McKay [b]

[a] *DComp – CCGT – UFSCar – Sorocaba, SP, Brazil*

[b] *Australian National University – Research School of Computer Science – Canberra, Australia*

Abstract

No snark has a 4-flow. A snark G is 4-edge-critical (or 4-vertex-critical) if, for every edge e (or pair of vertices (u, v)) of G, the graph obtained after contracting e (or identifying u and v) has a 4-flow. It is known that to determine whether a graph has a 4-flow is an *NP*-complete problem. In this paper, we present an improved exponential time algorithm to check whether a snark is 4-edge-critical (or 4-vertex-critical) or not. The use of our algorithm allowed us to determine the number of 4-edge-critical and 4-vertex-critical snarks of order at most 36.

Keywords: Snarks, Tutte's Flow Conjectures, Nowhere-zero k-flows

1 Introduction

Let $G = (V, E)$ be an undirected graph. A *k-edge-colouring* of G is an assignment of at most k colours to E such that no two adjacent edges are assigned the same colour. It is known that a cubic graph either has a 3-edge-colouring or a 4-edge-colouring (see [1, Theorem 17.4]). Holyer [8] showed that to determine whether a cubic graph has a 3-edge-colouring is an *NP*-complete problem.

A *snark* is, in essence, a cubic graph that does not admit a 3-edge-colouring. However, to avoid graphs that can be easily reduced to smaller

snarks, most authors define a *snark* as a cubic, cyclically 4-edge-connected graph with girth at least five and that does not admit a 3-edge-colouring. A graph is *cyclically k-edge-connected* if at least k edges must be removed in order to obtain at least two connected components with cycles. We will use the later definition.

Let D be an orientation of E and let f be a weight function that assigns integer weights to E. The *net-outflow* at vertex v, denoted $f(v)$, is the sum of the weights of the edges oriented leaving v minus the sum of the weights of the edges oriented entering v. A vertex v is said to be *balanced* if $f(v) = 0$. A pair of functions (D, f) is a *(nowhere-zero) k-flow* of G if none of the integer weights assigned by f is a multiple of k and every vertex of G is balanced. A *(nowhere-zero) mod k-flow* is defined similarly with the distinction that every vertex of G must be *balanced modulo k*, i. e. $f(v) \equiv 0 \pmod{k}$. Tutte showed that a graph G has a k-flow if and only if it has a mod k-flow (see [11, Proposition 3] for a proof). Tutte also proposed the following celebrated conjecture, which motivates our research.

Conjecture 1.1 (Tutte's 5-Flow Conjecture [10]) *Every bridgeless graph has a 5-flow.*

A famous theorem of Tutte states that a cubic graph has a 3-edge-colouring if and only if it has a 4-flow (see [1, Theorem 21.11] for a proof). It is known that a minimum counter-example to the 5-Flow Conjecture must be cubic and Jaeger [9, Theorem 9.3] showed that it must be a snark.

In 2008 [6], Silva and Lucchesi presented a study of *k-(flow)-edge-critical* and *k-(flow)-vertex-critical* graphs. A graph G without a k-flow is *k-edge-critical* (or *k-vertex-critical*) if, for every edge e (or for every pair (u, v) of vertices) of G, contraction of e (or identification of u and v) yields (without simplifying multiple edges) a graph that has a k-flow. Every k-edge-critical (or k-vertex-critical) graph has a $(k+1)$-flow (see [5]). Therefore, no 4-edge-critical (or 4-vertex-critical) snark is a counter-example to Tutte's 5-Flow Conjecture. Silva, Pesci and Lucchesi [7] observed that every snark has a 4-edge-critical snark as a minor. One approach towards resolving the 5-Flow Conjecture can be to attempt to extend to a non-4-edge-critical snark G the 5-flow of a 4-edge-critical snark minor H of G. As one first step in that direction Silva, Pesci and Lucchesi [7] identified the 4-edge-critical snarks of order at most 28, which was, until 2012, the largest database of snarks publicly available. In 2013, Brinkmann et al. [3] were able to generate a database of all snarks of order at most 36, a massive computational task, given the fact that the number of snarks grow exponentially with the order. The larger database can

be found in [2]. In this paper, we present the number of 4-edge-critical and 4-vertex-critical snarks in the larger database as well as the improved algorithm that allowed us to process the larger database despite its much greater size.

2 Algorithm and Results

As observed by Tutte, in order to determine whether a graph has a k-flow it suffices to determine whether it has a mod k-flow. Given a mod k-flow for a graph G, by reversing the orientation of some edge $e = (u, v)$ and complementing its weight $f(e)$ modulo k, the net-outflow at u is decreased by k and that of v is increased by k; thus resulting in another mod k-flow for G. Therefore, G has a mod k-flow for a given orientation D if and only if G has a mod k-flow for any orientation.

A (u, v)-*k-flow* is defined similarly to a mod k-flow with the difference that precisely two vertices, u and v, are not balanced modulo k (and the net-outflow at u and v complement each other). In [6, Theorems 3.1 and 3.2], Silva and Lucchesi observed that determining that a graph G without a k-flow is k-edge-critical is equivalent to finding a (u, v)-k-flow in G for every pair of adjacent vertices, and to finding a k-flow in $G - e$ for every edge e. Similarly, determining that G is k-vertex-critical is equivalent to finding a (u, v)-k-flow in G for every pair of vertices u, v, and to finding a k-flow in $G + (u, v)$ for every pair u, v (adding a new edge even if there is one there already).

The computer program due to Silva [4] used by Silva, Pesci and Lucchesi [7] for determining whether a snark is 4-vertex-critical is essentially a backtracking algorithm that fixes an orientation for the edges of the graph and recursively generates all possible weight functions checking for each one if precisely two vertices u and v are left unbalanced. If during the generation of all weight functions, a (u, v)-4-flow is found for every pair (u, v), the graph is 4-vertex-critical. Even though Silva's algorithm uses few pruning strategies, its perfomance was good enough in order to deal with the database of snarks of order at most 28. However, a rough estimate of the computer time needed to extend the computation to the database of snarks of order at most 36 using this same algorithm, even if 72 different instances were processed in parallel in a cluster, indicated nearly 6 months of computation. That observation lead us to investigate good pruning strategies to speed up the computation.

The third author observed that some simple pruning strategies shortened the total computation time to determine the number of 4-edge-critical snarks in this database to less than 2 days. Later, the first and second authors observed some extra simple pruning strategies that shortened the total compu-

tation time a little further. Both implementations were adapted to determine, among the 4-edge-critical snarks, which ones are 4-vertex-critical, which took around 2 hours for each implementation. In Table 1 we show the number of 4-edge-critical and 4-vertex-critical snarks of order at most 36. The results given by the two independent implementations agree. The description of these graphs can be found at [4].

order	10	18	20	22	24	26	28	30	32	34	36
snarks	**1**	**2**	**6**	**20**	**38**	**280**	**2.900**	**28.399**	**293.059**	**3.833.587**	**60.167.732**
4-edge-critical	**1**	**2**	**1**	**2**	**0**	**111**	**33**	**115**	**29**	**40.330**	**14.548**
4-vertex-critical	**1**	**2**	**1**	**2**	**0**	**111**	**33**	**115**	**13**	**40.328**	**13.720**

Table 1
Number of 4-flow-critical *snarks* for each order

Next we describe the improved algorithm implemented by the first and second authors (which is not, in essence, much different than the one implemented by the third author). Like Silva's algorithm, the improved algorithm is a backtracking algorithm that looks for a (u, v)-4-flow for several pairs (u, v), but it uses some effective pruning strategies. To begin with, unlike Silva's algorithm, our algorithm assumes that the input graph is a snark, therefore cubic. For a cubic graph, there are not many ways in which we can specify the weights of the edges incident to one particular vertex w so as to balance w modulo 4. First observe that the fixed orientation D must either direct all edges incident with w in one same direction or direct all but one in the same direction. For each of those cases, there are six combinations of weight assignments that balance w modulo 4, as depicted in Figure 1.

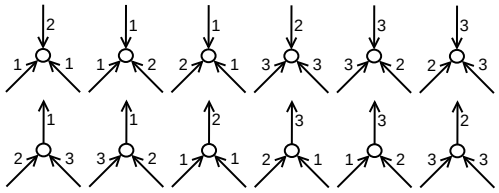

Figure 1. Weight possibilities

If the weight of one of the edges incident with w has already been fixed by the backtracking algorithm, then there are only two combinations of weight assigments to the remaining edges that balance w modulo 4; and if the weight of two edges of w have already been fixed, then either there is only one way to balance w or it cannot be balanced any longer.

Before triggering the backtracking algorithm that seeks a (u, v)-4-flow, we first specify an order O in which the vertices of $V - \{u, v\}$ must be tentatively balanced modulo 4. The order O specified is such that the next vertex to be balanced by the backtracking algorithm is one of the unbalanced vertices in $V - \{u, v\}$ with the least possible number of combinations of weight assignments that balance it. One such order O can be trivially found in polynomial time. As we attempt to balance vertices for which there are less combinations of weight assignments first, the breadth of the nodes in the search tree tends to be narrowed as well as the depth of some branches of the search tree.

The first vertex w in the order O is chosen arbitrarily and will be the only one with all three incident edges without fixed weights. Notice that by multiplying the weights of all edges of a (u, v)-4-flow by 3 modulo 4 we obtain another (u, v)-4-flow in which the edges of weight 3 become edges of weight 1 and vice-versa, while the edges of weight 2 do not change. Therefore, only the three leftmost combinations depicted in Figure 1 (which three depend on how the edges are directed) must be considered when we balance vertex w.

This improved backtracking algorithm to find a (u, v)-4-flow can be used to check if a snark is 4-edge-critical or 4-vertex-critical. To check for 4-edge-criticality of a snark is much faster than to check for 4-vertex-criticality as every cubic graph has just $O(n)$ edges. Since no non-4-edge-critical graph can be 4-vertex-critical, we first check for 4-edge-criticality in the whole database and only for the 4-edge-critical snarks do we check for 4-vertex-criticality.

Finally, observe that it may be possible to modify a (u, v)-4-flow so as to obtain a (u, y)-4-flow for some vertex y adjacent to v by edge e. If e is directed from v to y (or from y to v) and $f(e) - f(v) \not\equiv 0 \pmod 4$ (or $f(e) + f(v) \not\equiv 0 \pmod 4$) then it is possible to balance vertex v modulo k, and consequently unbalance y modulo k, by changing the value of $f(e)$. The result is a (u, y)-4-flow. A tree T_v, rooted at v, and containing all vertices y for which a (u, y)-4-flow can be obtained after a series of such modifications are applied can be computed in polynomial time. Similarly, a tree T_u, rooted at u, can be found. Once these trees are found, we know that for every pair of vertices (x, y) such that $x \in V(T_u)$, $y \in V(T_v)$ and $x \neq y$ there is a (x, y)-4-flow. Despite the extra (polynomial) cost in computing such trees, this process will potentially greatly reduce the number of executions of the exponential time backtracking algorithm. Experimentally, we observed that the computation of such trees nearly halved the total execution time to test for 4-vertex-criticality in the whole database.

3 Concluding Remarks

In this paper we described several simple pruning strategies that led to a considerable improvement of the exponential time algorithm to determine whether a snark is 4-edge-critical (or 4-vertex-critical). Such improvement allowed us to determine which snarks of order at most 36 are 4-edge-critical (or 4-vertex-critical) much faster than expected. We also discovered the first examples of 4-edge-critical snarks that are not 4-vertex-critical: there are 837 in total of order at most 36 and the smallest ones have order 32 (see Table 1). Having some database of 4-edge-critical and 4-vertex-critical snarks is an important first step towards the investigation of how non-critical snarks can be built from the critical ones and also of how the 5-flow of a critical snark may or may not be extended to a 5-flow of non-critical snarks built from it.

References

[1] Bondy, J. A. and U. S. R. Murty, "Graph Theory," Springer, 2008.

[2] Brinkmann, G., K. Coolsaet, J. Goedgebeur and H. Mélot, *House of graphs: A database of interesting graphs*, Discrete Applied Mathematics **161** (2013), pp. 311–314, Available at http://hog.grinvin.org.

[3] Brinkmann, G., J. Goedgebeur, J. Hägglund and K. Markström, *Generation and properties of snarks*, Journal of Combinatorial Theory, Series B **103** (2013), pp. 468–488.

[4] da Silva, C. N., *Personal homepage*, Available at http://www.dcomp.sor.ufscar.br/candida.

[5] da Silva, C. N. and C. L. Lucchesi, *Flow-critical graphs*, Technical report, Institute of Computing – University of Campinas – Brazil(2007).

[6] da Silva, C. N. and C. L. Lucchesi, *Flow-critical graphs*, Electronic Notes in Discrete Mathematics **30** (2008), pp. 165–170.

[7] da Silva, C. N., L. Pesci and C. L. Lucchesi, *Snarks and flow-critical graphs*, Electronic Notes in Discrete Mathematics **44** (2013), pp. 299–305.

[8] Holyer, I., *The NP-completeness of edge-coloring*, SIAM J. Comput. **10** (1981), pp. 718–720.

[9] Jaeger, F., *Nowhere-zero flow problems*, Selected Topics in Graph Theory **3**, Academic Press, 1988 pp. 71–95.

[10] Tutte, W. T., *A contribution to the theory of chromatic polynomials*, Can. J. Math. **6** (1954), pp. 80–91.

[11] Younger, D. H., *Integer flows*, J. Graph Theory **7** (1983), pp. 349–357.