Efficient (or less inefficient) computation of the zero forcing number

Louis Deaett, Quinnipiac University

Given a finite graph, call some vertices "filled" and then iteratively apply this rule until a stable state is achieved: When a filled vertex has only one unfilled neighbor, that neighbor becomes filled. The *zero forcing number* of the graph is the smallest number of initially filled vertices for which this process terminates with all vertices filled. To determine this value for a general graph is an NP-hard problem. Even so, there is a need for algorithms that can do this for graphs that are not too large while using a reasonable amount of time and storage. One example is the "wavefront algorithm" implemented in software released in 2009 by Grout, Hall, and Lagrange. We review how this algorithm works and introduce a novel, related algorithm. We give examples of the relative performance of this new algorithm, discuss how its performance could be improved further, and consider how it may be adapted to compute other variants of the zero forcing number.

Keywords: zero forcing number, graph, algorithm