PACE Solver Description: HeiTwin (Exact)

Dennis Jakob
Heidelberg University, Heidelberg, Germany

Ernestine Großmann
Heidelberg University, Heidelberg, Germany

Nikita-Nick Funk
Heidelberg University, Heidelberg, Germany

Thomas Möller
Heidelberg University, Heidelberg, Germany

Abstract

HeiTwin-Exact implements graph reductions and a branch and bound algorithm to compute a contraction sequence for the twin-width of the graph. It uses different heuristic strategies to compute an upper bound, and looks for chordless cycles as a lower bound if the heuristic twin-width is 2. To break symmetry, two independent consecutive contractions are done in a fixed order.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases twin-width, exact, branch and bound, reductions

1 Introduction

The concept of *twin-width* was introduced by Bonnet et al. in 2021 [3]. Intuitively, it describes how much a graph differs from a cograph. Not much is known regarding twin-width other than it being NP-complete to determine whether a given graph has a bounded twin-width of at most four [1]. It is also NP-hard to approximate the twin-width with an approximation ration better than 5/4 [1]. This work proposes a branch-and-bound approach to quickly find an exact contraction sequence for a given graph.

2 Preliminaries

A *trigraph* is an *undirected, simple* graph $G = (V, E)$ where the edge set $E(G)$ is partitioned into the sets $B(G)$ of black edges and $R(G)$ of red edges. A regular graph can be considered a trigraph for which $E(G) = B(G)$ and $R(G) = \emptyset$. The set $N_G(v)$ of neighbors of a vertex $v \in V(G)$ are all vertices adjacent to v by a black or red edge. A neighbor $u \in N_G(v)$ is called a *black neighbor* of v if $uv \in B(G)$ and a *red neighbor* if $uv \in R(G)$. The *red degree* of a vertex $v \in V(G)$ is the number of red neighbors of v. A *d-trigraph* is a trigraph where each vertex has a red degree of at most d.

A trigraph G' is obtained from a trigraph G by *contracting* two, not necessarily adjacent vertices. Two vertices u and v are contracted by merging them into a new vertex w and updating the edges as follows: Every vertex in the symmetric difference $N_G(u)\Delta N_G(v)$ is made a red neighbor of w. If a vertex $x \in N_G(u) \cap N_G(v)$ is both a black neighbor of u and of v it is made a black neighbor of w, otherwise x is made a red neighbor of w. All other edges remain unchanged. A *d-sequence* or *sequence of d-contractions* of a graph G is a sequence of d-trigraphs $G_0, G_1, \cdots, G_{n-1}$ such that $G_0 = G$ and G_{n-1} is the graph on a single vertex. For $i \geq 1$ G_i is obtained from G_{i-1} by contraction. The twin-width *tww(G)* is the smallest integer d for which G admits to such a d-sequence.

Two contractions of a graph G_i are called independent if it does not matter in which order they are done. That is, if the maximum red degree in the contraction sequence and the resulting graph G_{i-2} are the same.

3 Solver Overview

For computing the exact twin-width we implemented a branch and bound algorithm. First, exact reduction rules, as described in Section 4, are applied to reduce the initial graph size. An initial upper bound is computed by using multiple heuristic solutions and picking the best one. The heuristics are described in Section 5.1. In case the heuristic twin-width is 2, we check if we can also find a lower bound of 2 to prove that the heuristic solution is exact. Afterwards, a branch and bound algorithm checks the remaining search space, updating the current best solution, using this as an upper bound. The search space is reduced by breaking symmetry in consecutive contractions, doing independent contractions in a fixed order.

4 Reductions

The complexity of the branch and bound algorithm grows exponentially with respect to the number of vertices of the graph. Reducing the number of vertices is therefore necessary to speed up the algorithm. To reduce the problem size connected components, complement graphs and twins are used. This is done once at the beginning.

4.1 Connected Components

If the graph is not connected, the twin-width can be computed for every connected component separately. The twin-width of the graph is the maximum of all twin-widths of the components [4]. The remaining vertices can be contracted in any order as there are no edges between them.

4.2 Complement Graph

The twin-width can also be computed on the complement graph. If the complement graph is disconnected, the first reduction rule can be applied again and the connected components can be solved independently. Each component can be solved on the original graph or the complement graph. The one of smaller size is often the better choice, as the complexity of the branch and bound algorithm also depends on the number of edges.

4.3 Twins

Two vertices are twins if they have the same neighborhood, excluding themselves. Contracting them is an exact reduction as follows from Theorem 1.

▶ **Theorem 1.** *Contracting twins where the edge colors coincide, that is twins which have the same red and black edges, is an exact reduction.*

Proof. After contracting a pair of twins, any following contraction can only lead to the same or a lower red degree: after such a twin contraction, for all vertices the edge colors remain the same and the neighborhood is either the same as before the contraction or it was reduced by one (which is one of the twins which was contracted). ◀

Vertices of degree zero or one can be checked in linear time by going over all vertices once. If the first vertex of degree zero is found, it is remembered and every time another vertex of degree zero is found, it is contracted with the remembered one. Twins of degree one can be found similarly, but if a vertex of degree one is found it is stored in an array at the position of its neighbor. If a vertex is already stored there and another one is found, they are twins and can be contracted.

5 Branch & Bound

The basis of the branch and bound algorithm is an exhaustive search approach. The idea of the algorithm is to iterate over every feasible solution and choose the best one. In the case of twin-width computation, a feasible solution is any contraction sequence that contracts two vertices and contracts the graph into a single vertex. Therefore, we need to check every possible contraction sequence. We do this in a DFS-manner by always fixing one contraction and going to the next one. If the graph is contracted into a single vertex, we have a feasible solution. If the twin-width is lower than the current best solution, we store this contraction sequence as the new current best one. Then, the last contraction is undone and a different contraction is picked, assuming there exists a contraction that has not been considered before. Otherwise, we keep uncontracting until there is an unconsidered contraction or the twin-width of all possible contraction sequences was checked, in which case we are done.

The exhaustive search algorithm is extended to a branch and bound algorithm by reducing the search space through bounds. The search space can be pruned by an upper bound (if a partial solution is already known to be non-optimal) and a lower bound (if the current best solution is known to be optimal). To reduce the symmetry in the branch and bound algorithm two independent contractions are only done in one order.

5.1 Upper and Lower Bound

To improve the efficiency of the branch and bound algorithm a good upper bound is very important. The twin-width of any feasible solution yields an upper bound. Assume that a solution with twin-width x was already found. If at some point in the contraction sequence there exists a vertex with at least x red edges, no better solution can exist, because the twin-width is the highest red degree in a contraction sequence and further contractions can only increase the twin-width. Therefore, this branch can be skipped, meaning the last contraction can be undone without contracting the graph into a single vertex. An initial upper bound is computed by running different heuristic strategies including:

- *RedDegLimit*: Start with a limit of 0, contract all vertex pairs that do not exceed this limit (in the order as they appear in the graph), increment the limit and repeat until the graph is contracted.
- *RedDegLimitRandom*: Similar to *RedDegLimit*, but contractions are done in a random order. Experiments showed that 25 repetitions are able to find a better solution in some cases while still not needing too much time.
- *Greedy*: Greedily perform contractions according to the maximum red degree in the neighborhood. This is only done for graphs with at most 1000 vertices due to the increasing runtime.
- *DepthLimitedBB*: Iteratively do a branch and bound algorithm with a depth limit until the graph is contracted. This is done for different depth limits from 3 to 10 depending on the order of the graph.

If it is possible to compute a lower bound and there exists a solution which abides by this lower bound, this is an optimum solution and the search can be stopped. However, this bound has limited applicability. It is NP-complete to decide whether a graph has a twin-width of at most 4 [1]. It is possible to check whether the twin-width is at most 1 in polynomial time [2], and there are some patterns which lead to a twin-width of at least 2. One such pattern is a chordless cycle of length at least 5 [2], which is implemented by using a modified BFS. If a heuristic solution of twin-width 2 can be found, an attempt to find a chordless cycle is made. If one is found the branch and bound algorithm is skipped entirely.

References

1 Pierre Bergé, Édouard Bonnet, and Hugues Déprés. Deciding twin-width at most 4 is np-complete. *CoRR*, abs/2112.08953, 2021. URL: `https://arxiv.org/abs/2112.08953`, `arXiv:2112.08953`.

2 Édouard Bonnet, Eun Jung Kim, Amadeus Reinald, Stéphan Thomassé, and Rémi Watrigant. Twin-width and polynomial kernels. *CoRR*, abs/2107.02882, 2021. URL: `https://arxiv.org/abs/2107.02882`, `arXiv:2107.02882`.

3 Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. *J. ACM*, 69(1):3:1–3:46, 2022. `doi:10.1145/3486655`.

4 André Schidler and Stefan Szeider. A SAT approach to twin-width. *CoRR*, abs/2110.06146, 2021. URL: `https://arxiv.org/abs/2110.06146`, `arXiv:2110.06146`.