A Hierarchical Sparsification Technique for Faster Algorithms in Graphs and Game Graphs*

Veronika Loitzenbauer (vl@cs.univie.ac.at)

University of Vienna, Faculty of Computer Science, Austria

We present a sparsification technique, called *hierarchical graph decomposition*, and survey its recent applications to problems relevant in computer-aided verification. We provide some intuition for which kind of problems this technique can lead to algorithms with improved asymptotic runtimes, in particular for dense graphs. Apart from classical graphs, we also consider Markov decision processes (MDPs) and (two-player) game graphs. For the definitions of the listed problems as well as their application in computer-aided verification we refer the reader to the cited literature. For example, parity games with three priorities can be used to analyze timed automaton games (a model for real-time systems) with reachability and safety objectives [9, 8, 4, 7].

Related Work. Henzinger et al. [10] introduced the hierarchical graph decomposition as a graph sparsification technique to replace 'm', the number of edges, with 'n', the number of vertices, in the running time bound. They considered the problem of quickly identifying a new connected component in an *undirected graph* after a batch of *edge deletions*, motivated by a problem in computational biology. Chatterjee and Henzinger [2] extended the technique to *directed graphs*, *MDPs*, and *game graphs*. They applied it to two problems where the basic algorithms are based on repeated *vertex deletions*: Computing the maximal end-component decomposition of MDPs (MEC) and computing the winning sets of both players in Büchi games (Büchi).

problem	previous runtime	when $m = \Theta(n^2)$	new algorithm
MEC	$O(\min\{mn^{2/3}, m^{3/2}\})$ [1]	$O(n^{8/3})$	$O(n^2)$ [2]
Büchi (ignoring log factors)	$O(mn)$ [6, 5]	$O(n^3)$	$O(n^2)$ [2]
Parity-3	$O(mn)$ [14]	$O(n^3)$	$O(n^{5/2})$ **[3]**
Streett (simplified runtime)	$O(\min\{mn, m^{3/2}\})$ [12]	$O(n^3)$	$O(n^2)$ **[3]**
2SCC	$O(mn)$ [15, 13]	$O(n^3)$	$O(n^2)$ **[11]**

Results. In recent work [3] we applied the technique in two ways: We showed how the runtime analysis can be modified in the case where vertices are not removed from the

*This abstract is based on joint work [3, 11] with Krishnendu Chatterjee, Monika Henzinger, and Sebastian Krinninger. This work was partially funded by the Austrian Science Fund (FWF): P23499-N23 and the Vienna Science and Technology Fund (WWTF) grant ICT10-002. Additionally, the research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement no. 340506.

graph, which yields a faster algorithm for the nonemptiness problem of Streett automata (STREETT) in dense graphs; and we showed how the technique can be used to combine and speed up known algorithms for the winning sets in Parity games with three priorities (PARITY-3). The latter approach extends to Parity games with an arbitrary number of priorities and is different from the algorithm for Büchi games (which are equal to Parity games with two priorities). For completeness we also list our recent improved algorithm for a classical graph problem (2SCC) [11]. We summarize the running time improvements in our and related work in the table above. Note that while for some problems the runtime can only be improved for dense graphs, for other problems the runtime is improved for all cases except when the number of edges m is in the order of the number of vertices n.

Decomposition. We decompose a directed graph $G = (V, E)$ with $m = |E|$ edges and $n = |V|$ vertices into a hierarchy of graphs $G_i = (V, E_i)$ for levels $i \in \{1, \ldots, \lceil \log n \rceil\}$. We will define the set of edges E_i such that $|E_i|$ is $O(n \cdot 2^i)$, $E_{i-1} \subseteq E_i$ for all $i > 1$, and $E_{\lceil \log n \rceil} = E$. Whether an edge $(u, v) \in E$ is included in E_i will depend on the in- and out-degree of u and v, and can for game graphs additionally depend on the player to which the vertices on the other end of the incoming or outgoing edges of u and v belong. For example, in the algorithms for MEC and STREETT the set E_i contains all edges $(u, v) \in E$ for which the out-degree of u is at most 2^i. In the algorithm for PARITY-3 the set E_i additionally contains the first 2^i incoming edges of each vertex, where the incoming edges are sorted such that those from Player 2 come first. For now we call all vertices that are *not* missing outgoing edges in G_i *white.*

Size-Degree Relation. To see why this definition can be useful, think about a strongly connected component that has no outgoing edges, called a *bottom* SCC. Let C be a bottom SCC with 2^i vertices. Then each vertex in C must have an out-degree of less than 2^i, i.e., in G_i all vertices in C are white. This implies that if we search for bottom SCCs in G_i that only contain white vertices, then we will detect C. In the algorithms for MEC and STREETT we repeatedly search for such bottom SCCs after some vertices were removed from the graph. This search is started at level $i = 1$, and the level is increased by one until the search is successful. Whenever we have to go up to level i^* to identify such a bottom SCC C, then C has to contain more than 2^{i^*-1} vertices since otherwise it would have been identified at level $i^* - 1$. The hierarchical graph decomposition can only be applied if one can show a similar relation between the size of searched sets and the degree of the vertices in this set. For Parity games the searched sets of vertices are part of the winning set of one of the players.

Runtime Analysis. A bottom SCC in G_i can be found in time $O(|E_i|)$, which is $O(n \cdot 2^i)$. In the algorithm for MEC the vertices in the identified bottom SCC are then removed from the graph and thus can be charged the work to identify them, which leads to a total runtime of $O(n^2)$. Although for STREETT the identified bottom SCC is not removed from the graph, we can show the same runtime (for this part of the algorithm) using a parallel search in the reverse graph. For PARITY-3 the time to identify a part of the winning set is proportional to $|E_i|$ times the size of this part; the runtime bound of $O(n^{2.5})$ comes from searching only for parts with at most $\sqrt{n}$ vertices with the hierarchical graph decomposition and using an $O(n^2)$ algorithm for larger parts.

When can it be applied? All the mentioned problems can be seen as vertex or edge partitioning problems (e.g. partition of vertices into winning set of Player 1 and (parts of) winning set of Player 2). Further, they all have a basic algorithm that iteratively refines a maintained partition, i.e., there exists a "fast" way to either further divide a set in the maintained partition or to decide that no further refinement is needed. When, e.g., the algorithm for STREETT identifies a bottom SCC, then it recurses on each of the bottom SCC and the subgraph induced by the remaining vertices and thereby refines the (implicitly) maintained partition. The crucial step in applying the presented technique is to show a similar size-degree relation, as described above for bottom SCCs, for a set of vertices or edges that can be used to refine the maintained partition and that can be found "fast" (e.g. in linear time). We believe that more problems with such a structure exist.

References

[1] K. Chatterjee and M. Henzinger. Faster and Dynamic Algorithms For Maximal End-Component Decomposition And Related Graph Problems In Probabilistic Verification. In *SODA*, pages 1318–1336, 2011.

[2] K. Chatterjee and M. Henzinger. Efficient and Dynamic Algorithms for Alternating Büchi Games and Maximal End-component Decomposition. *Journal of the ACM*, 61(3):15, 2014. announced at SODA'11 and SODA'12.

[3] K. Chatterjee, M. Henzinger, and V. Loitzenbauer. Improved Algorithms for One-Pair and k-Pair Streett Objectives. http://arxiv.org/abs/1410.0833v2, October 2014.

[4] K. Chatterjee, T. A. Henzinger, and V. S. Prabhu. Timed parity games: Complexity and robustness. *Logical Methods in Computer Science*, 7(4), 2011.

[5] K. Chatterjee, T.A. Henzinger, and N. Piterman. Algorithms for Büchi games. In *Games in Design and Verification (GDV)*, 2006.

[6] K. Chatterjee, M. Jurdziński, and T.A. Henzinger. Simple stochastic parity games. In *CSL'03*, volume 2803 of *LNCS*, pages 100–113. Springer, 2003.

[7] K. Chatterjee and V. S. Prabhu. Synthesis of memory-efficient, clock-memory free, and non-zeno safety controllers for timed systems. *Inf. Comput.*, 228:83–119, 2013.

[8] L. de Alfaro and M. Faella. An accelerated algorithm for 3-color parity games with an application to timed games. In *CAV*, pages 108–120, 2007.

[9] L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR*, pages 142–156, 2003.

[10] M. Henzinger, V. King, and T. Warnow. Constructing a Tree from Homeomorphic Subtrees, with Applications to Computational Evolutionary Biology. *Algorithmica*, 24:1–13, 1999.

[11] M. Henzinger, S. Krinninger, and V. Loitzenbauer. 2-edge and 2-vertex strongly connected components in quadratic time. http://arxiv.org/abs/1412.6466, December 2014.

[12] M. Henzinger and J.A. Telle. Faster Algorithms for the Nonemptiness of Streett Automata and for Communication Protocol Pruning. In *SWAT*, pages 16–27, 1996.

[13] R. Jaberi. On computing the 2-vertex-connected components of directed graphs. http://arxiv.org/abs/1401.6000v1, January 2014.

[14] M. Jurdziński. Small Progress Measures for Solving Parity Games. In *STACS*, pages 290–301, 2000.

[15] H. Nagamochi and T. Watanabe. Computing k-edge-connected components of a multigraph. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, E76–A(4):513–517, 1993.