A Parametrized Analysis of Algorithms on Hierarchical Graphs

Rachel Faran(✉) and Orna Kupferman

School of Engineering and Computer Science, Hebrew University, 91904 Jerusalem, Israel
rachelmi@cs.huji.ac.il

Abstract. *Hierarchical graphs* are used in order to describe systems with a sequential composition of sub-systems. A hierarchical graph consists of a vector of subgraphs. Vertices in a subgraph may "call" other subgraphs. The reuse of subgraphs, possibly in a nested way, causes hierarchical graphs to be exponentially more succinct than equivalent flat graphs. Early research on hierarchical graphs and the computational price of their succinctness suggests that there is no strong correlation between the complexity of problems when applied to flat graphs and their complexity in the hierarchical setting. That is, the complexity in the hierarchical setting is higher, but all "jumps" in complexity up to an exponential one are exhibited, including no jumps in some problems.

We continue the study of the complexity of algorithms for hierarchical graphs, with the following contributions: (1) In many applications, the subgraphs have a small, often a constant, number of exit vertices, namely vertices from which control returns to the calling subgraph. We offer a parameterized analysis of the complexity and point to problems where the complexity becomes lower when the number of exit vertices is bounded by a constant. (2) We describe a general methodology for algorithms on hierarchical graphs. The methodology is based on an iterative compression of subgraphs in a way that maintains the solution to the problems and results in subgraphs whose size depends only on the number of exit vertices, and (3) We handle labeled hierarchical graphs, where edges are labeled by letters from some alphabet, and the problems refer to the languages of the graphs.

1 Introduction

Systems are typically constructed in a compositional manner. The two basic types of compositions are *concurrent* and *sequential*. In a concurrent composition, the state space of the composed system is essentially the product of the state spaces of its underlying components. In a sequential composition, the state

The research leading to these results has received funding from the European Research Council under the European Union's 7th Framework Programme (FP7/2007-2013, ERC grant no. 278410).

G. Pighizzini and C. Câmpeanu (Eds.): DCFS 2017, LNCS 10316, pp. 114–127, 2017.
DOI: 10.1007/978-3-319-60252-3_9

space of the composed system consists of copies of the state spaces of its underlying components. Since a component may be reused several times, in particular when nesting is allowed, both types of compositions allow an exponentially more succinct presentation of systems [1]. A natural question is whether one can reason about the succinct presentation in order to answer questions about the composition.

Beyond the theoretical interest in studying this question, the challenge of reasoning about systems in their succinct presentation is of great importance in the context of formal verification. There, we reason about a hardware or software system by translating it to a finite state machine (FSM) [5]. The FSM is a labeled graph, and since it lacks the internal structure of the high-level description of the system, we refer to it as a *flat graph*. The exponential blow-up in the translation of the system to a flat graph is typically the computational bottleneck in model-checking algorithms. For *concurrent graphs*, where components are composed in a concurrent manner [8], there has been extensive research on compositional model checking (cf., [6]). Compositionality methods are successfully applied in practice, but it is a known reality that they cannot always work. Formally, the system complexity of the model-checking problem (that is, the complexity in terms of the system, assuming a specification of a fixed length) for all common temporal logics is exponentially higher in the concurrent setting [14]. This exponential gap is carried over to other related problems such as checking language-containment and bisimulation—all are exponentially harder in the concurrent setting [11].

For sequential compositions, which are common in software systems (reuse of components amounts to calling a procedure), the model used is that of *hierarchical graphs* (HGs, for short). An HG consists of a vector of subgraphs. A subgraph may be used several times in the composition. Technically, this is done via special vertices, termed *boxes*, that are substituted in the composition by subgraphs. Each subgraph has an entry vertex and a set of exit vertices. In order to ensure a finite nesting depth of substitutions, the subgraphs are indexed, and a box of a graph can only *call* (that is, be substituted by) subgraphs with a strictly bigger index. A naive approach to model checking a system with such a sequential composition is to "flatten" its HG by repeatedly substituting references to subgraphs by copies of these subgraphs. This, however, results in a flat graph that is exponential in the nesting depth of the hierarchical system. In [2], it is shown that for LTL model checking, one can avoid this blow-up altogether, whereas for CTL, one can trade it for an exponential blow-up in the (often much smaller) size of the specification and the maximal number of exits of sub-structures. Likewise, μ-calculus model-checking in hierarchical systems is only PSPACE-complete [3].

For general graph algorithms, there is a good understanding that the exponential succinctness of concurrent graphs comes with a computational price. For example, classic NEXPTIME-complete problems (cf., the succinct-Hamiltonian-path problem) are the succinct versions of NP-complete problems [10]. A similar exponential gap exists in other complexity classes. For HGs, the picture is less clear. Indeed, it is shown in [16] that there is no strong correlation between

the complexity of problems when applied to flat graphs and their complexity in the hierarchical setting. More specifically, Lengauer and Wagner examine a large number of problems in different complexity classes, and show that while for some, the hierarchical setting makes the problem exponentially harder, for some it does not: problems that are in LOGSPACE, NLOGSPACE, PTIME, NPTIME, and PSPACE in the flat setting, may stay in this complexity or jump to any (at most exponentially harder) class in the hierarchical setting. For example, graph reachability is NLOGSPACE-complete for flat graphs and is PTIME-complete for HGs. On the other hand, alternating graph reachability (that is, where the graph is a two-player game, and the players alternate moves picking a successor vertex) is PTIME-complete for flat graphs and is PSPACE-complete for HGs. Additional examples can be found in [13,15,17].

We continue to investigate the complexity of algorithms on HGs. Our contributions are described below.

Parameterized Analysis. In many applications, the subgraph components have a small, often a constant, number of exit vertices. Indeed, these exit vertices form the interface of a procedure or a library component. We offer a *parameterized analysis* of the complexity of algorithms on HGs [7]. In particular, while [16] proves that the problems *Hamiltoniam path*, *3-colorability*, and *independent set* are PSPACE-complete for hierarchical graphs, we show that they are in NP, namely as hard as in the flat setting, for *HGs with a constant number of exits* (CE-HGs, for short). Likewise, while the problem of finding a *maximal flow* in a network jumps from PTIME to PSPACE in the hierarchical setting, we conjecture that it is in PTIME for CE-HGs, and prove that this is indeed the case for HGs with at most 3 exits. As another example, while [16] proves that *alternating reachability* is PSPACE-complete for HGs, we prove it is in PTIME for CE-HGs. We note that analyzing the complexity of the PSPACE algorithms in [16] for a constant number of exit vertices does not lead to improved upper bounds. Thus, our tighter complexity results involve new algorithms, as described below.

Methodology. We define a general methodology for algorithms on HGs. For a problem P, we say that a function f on subgraphs *maintains* P if for every graph G, the graph $f(G)$ has the same entry and exit vertices as G, and every HG that calls G may call $f(G)$ instead without influencing P. Consider, for example, the problem P of finding a shortest path between two vertices in a graph. Consider also the function that maps a subgraph G with entry vertex s and set T of exit vertices to a tree whose vertices are $\{s\} \cup T$, whose edges are $\{s\} \times T$, and in which the weight of an edge $\langle s, t\rangle$ is the length of the shortest path from s to t in G. It is easy to see that f maintains P. Moreover, the size of $f(G)$ depends only on the number of exit points in G. For two complexity functions g_{size} and g_{time}, we say that a problem P is (g_{size}, g_{time})-*compressible* if there is a function f that maintains P, maps graphs G with a set T of exit vertices to graphs whose size is bounded by $g_{size}(|T|)$, and does so in time $g_{time}(|G|, |T|)$. We use the notion of compressibility in order to develop algorithms on HGs that iteratively replace boxes that call internal subgraphs by the compressed version of these subgraphs. In particular, when g_{size} and g_{time} are polynomial, then the

complexity of solving P in HGs adds a polynomial factor to the complexity of its solution in flat graphs. Moreover, the fact g_{time} has two parameters, enables the parameterized complexity analysis. In particular, when the number of exit vertices is bounded by a constant, the complexity is induced by g_{time} only. Thus, if g_{time} is linear (or polynomial) in its first parameter, then so is the complexity of solving P in CE-HGs. We extend the notion of compressibility to nondeterministic functions f, where compression of graphs also generates a witness to the soundness of the compression. The witness can be verified in time that is polynomial in the number of exit vertices, and thus nondeterministic compression is used for obtaining NP upper bounds in the hierarchical setting.

Labeled Graphs. We study *labeled hierarchical graphs*, where each edge in the graph is labeled by a letter from some alphabet. In [1], the authors study *communicating hierarchical state machines*, which are similar to labeled hierarchical graphs. The study in [1] extends classical decision problems from automata theory to the hierarchical setting. In particular, the authors study reachability, emptiness, and language inclusion. Here, we study an extension of classical graph-theory problems to the labeled setting. The input to such problems includes, in addition to the graph, a specification that constrains the paths in the graph. For example, [4] studies the problem of finding a shortest path that satisfies regular and even context-free specifications, and shows that the problem stays in polynomial time. On the other hand, research in regular path queries shows that the problem of finding a shortest simple path that satisfies a regular specification is NP-complete [19]. Typical algorithms on labeled graphs are based on a *product* between the graph and an *automaton* for the specification. We show how the compression of subgraphs defined above can be extended to products of labeled HGs and automata, and demonstrate the use of such a compression.

Due to lack of space, the study of labeled graphs as well as some of the proofs are omitted, and can be found in the full version, in the authors' URLs.

2 Preliminaries

A *graph with source and target vertices* is $G = \langle V, s, T, E\rangle$, where V is a set of vertices, $s \in V$ is a source vertex, $T \subseteq V$ is a set of target vertices, and $E \subseteq V \times V$ is an edge relation. A graph may be weighted, in which case the tuple G includes also a weight function $w : E \to \mathbb{R}^+$.

A *hierarchical graph* (HG, for short) consists of a vector of subgraphs that together compose a graph. A subgraph may be used several times in the composition. Technically, this is done via special vertices, called *boxes*, that are substituted in the composition by other subgraphs. In order to ensure a finite nesting depth of substitutions, the subgraphs are indexed, and a box of a graph can only *call* (that is, be substituted by) subgraphs with a strictly bigger index. Formally, an HG $\mathcal{G}$ is a tuple $\langle G_1, \ldots, G_n\rangle$, where each subgraph is $G_i = \langle V_i, B_i, in_i, Exit_i, \tau_i, E_i\rangle$, where V_i and B_i are sets of vertices and boxes, respectively. We assume that $B_n = \emptyset$ and that $V_1, \ldots, V_n, B_1, \ldots, B_{n-1}$ are pairwise disjoint. Then, $in_i \in V_i$ is an entry vertex for G_i, and $Exit_i \subseteq V_i$ is a set of

exit vertices for G_i. The function $\tau_i : B_i \to \{i+1, \ldots, n\}$ maps each box of G_i to an index greater than i. If $\tau_i(b) = j$, we say that the box b is *substituted* by G_j in G_i. Finally, E_i is an edge relation. Each edge in E_i is a pair (u,v) with source u and target v. The source u is either a vertex of G_i, or a pair $\langle b,x \rangle$, where $b \in B_i$, and $x \in Exit_{\tau_i(b)}$. That is, u may be a box b coupled with an exit vertex of the subgraph by which b is about to be substituted. The target v is either a vertex or a box of G_i. Formally, $E_i \subseteq (V_i \cup (\cup_{b \in B_i} \{b\} \times Exit_{\tau_i(b)})) \times (V_i \cup B_i)$. We refer to $\{in_i\} \cup Exit_i$ as the set of *interface vertices* of G_i, denoted V_i^{face}. We refer to G_n as the *bottom subgraph* of $\mathcal{G}$.

When $|Exit_i|$ is bounded by a constant for all $1 \leq i \leq n$, we say that $\mathcal{G}$ is a *hierarchical graph with constant number of exit vertices* (CE-HG, for short). A *weighted HG* is an HG with weight functions $w_i : E_i \to \mathbb{R}^+$ for all $1 \leq i \leq n$. Note that weights are associated with edges (rather than vertices).

A subgraph without boxes is *flat*. Each HG can be transformed to a flat graph, referred to as its *flat expansion*, by recursively substituting each box by a copy of the suitable subgraph. Formally, given an HG $\mathcal{G}$, for each subgraph G_i we inductively define its flat expansion $G_i^f = \langle V_i^f, in_i, Exit_i, E_i^f \rangle$, where $V_i^f = V_i \cup (\cup_{b \in B_i} \{b\} \times V^f_{\tau_i(b)})$. Note that different boxes in G_i can be substituted by the same subgraph. By substituting each box b by a set of vertices $\{b\} \times V^f_{\tau_i(b)}$, we preserve b as an identifier. The edge relation E_i^f includes the following edges:

- (u, v) such that $u, v \in V_i$ and $(u, v) \in E_i$,
- $(u, \langle b, v \rangle)$ such that $u \in V_i$, $v = in_{\tau_i(b)}$ and $(u, v) \in E_i$,
- $(\langle b, u \rangle, v)$ such that $u \in Exit_{\tau_i(b)}$, $v \in V_i$ and $(u, v) \in E_i$, and
- $(\langle b, u \rangle, \langle b, v \rangle)$ such that $u, v \in V^f_{\tau_i(b)}$ and $(u, v) \in E^f_{\tau_i(b)}$.

The graph G_1^f is the flat expansion of $\mathcal{G}$, and we denote it by $\mathcal{G}^f$.

Example 1. In Fig. 1 we describe an HG $\mathcal{G} = \langle G_1, G_2 \rangle$, where G_1 includes two boxes, b_1 and b_2, with $\tau_1(b_1) = \tau_1(b_2) = 2$. The bottom subgraph G_2 is flat. The flat graph $\mathcal{G}^f$ is described on the right. □

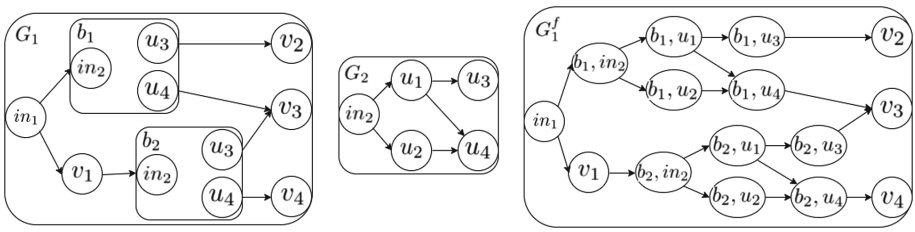

Fig. 1. A hierarchical graph

We define the *size* of a graph $G = \langle V, s, T, E\rangle$, denoted $|G|$, as $|V| + |E|$. For an HG $\mathcal{G} = \langle G_1, \ldots, G_n\rangle$, we define the size of each subgraph $G_i = \langle V_i, B_i, in_i, Exit_i, \tau_i, E_i\rangle$, denoted $|G_i|$, as $|V_i| + |B_i| + |E_i|$. Note that the sizes of the subgraphs that are substituting the boxes in B_i do not affect $|G_i|$. We denote by $|\mathcal{G}|$ the size of all of the subgraphs in $\mathcal{G}$, namely $|G_1| + \cdots + |G_n|$.

It is not hard to see that the hierarchical setting is exponentially more succinct. Formally, we have the following.

Proposition 1 [1]. *Flattening an HG may involve an exponential blow up. That is, $\mathcal{G}^f$ may be exponentially larger than $\mathcal{G}$. The exponential blow-up applies already to the diameter of the graph, and applies even when all the subgraphs in $\mathcal{G}$ have a single exit vertex.*

3 Compression of Hierarchical Graphs

Let $\mathscr{G}$ be the set of all flat graphs with source and target vertices. Let P be a problem on graphs. That is, $P : \mathscr{G} \to \{0, 1\}$ may be a decision problem, say deciding whether all the vertices in T are reachable from s, or $P : \mathscr{G} \to \mathbb{R}$ may be an optimization problem, say finding the length of a shortest path from s to T. For an HG $\mathcal{G}$, solving P on $\mathcal{G}$ amounts to solving P in G_1^f, namely in the flat expansion of $\mathcal{G}$. A naive way to do so is to construct the flat expansion of $\mathcal{G}$ and then solve P on it. By Proposition 1, this may involve an exponential blow-up. In this section we present a general methodology for reasoning about HGs without generating their flat expansions. Essentially, we iteratively replace subgraphs, starting from the innermost ones, by compressed flat versions in a way that maintains $P(\mathcal{G})$. The important fact is that the size of the compressed versions depends only on the number of exit vertices, which enables an analysis of the parametrized complexity of solving P in HGs.

We now formalize this intuition. Consider an HG $\mathcal{G} = \langle G_1, \ldots, G_n\rangle$. A *bottom flatenning* of $\mathcal{G}$, denoted $\mathcal{G}\downarrow$, is the HG obtained from $\mathcal{G}$ by removing G_n and replacing all boxes b that call G_n by G_n. Formally, $\mathcal{G}\downarrow = \langle G'_1, \ldots, G'_{n-1}\rangle$, where for every $1 \leq i \leq n-1$, the subgraph G'_i is obtained from G_i by replacing every box $b \in B_i$ with $\tau_i(b) = n$ by a copy of G_n. As in the case of a flat expansion, the copies preserve b as an identifier. Let $G_i = \langle V_i, B_i, in_i, Exit_i, \tau_i, E_i\rangle$, and let $B_i^n = \{b \in B_i : \tau_i(b) = n\}$. Then, $G'_i = \langle V_i \cup (B_i^n \times V_n), B_i \setminus B_i^n, in_i, Exit_i, \tau_i, E_i \cup (B_i^n \times E_n)\rangle$, where $B_i^n \times E_n$ includes the copies of E_n in the different substitutions. Formally, $\langle u, u'\rangle \in E_n$ iff for all $b \in B_i^n$, we have $(\langle b, u\rangle, \langle b, u'\rangle)$. Note that transitions in E_i that involve a box b with $\tau_i(b) = n$ have end-points in $\langle b, in_n\rangle$ and $\{b\} \times Exit_n$, so the transitions in G'_i are well defined eventhough we remove b.

While $\mathcal{G}\downarrow$ has one less subgraph, the boxes in its subgraphs that call G_n have been replaced by G_n, so $|\mathcal{G}\downarrow| \geq |\mathcal{G}|$. The key idea behind our methodology is that often we can replace G_n by a subgraph G' of a constant size that retains its properties. We now formalize this intuition. Consider the HG $\mathcal{G} = \langle G_1, \ldots, G_n\rangle$. Given a graph with source and target vertices $G' = \langle V', s, T, E'\rangle$, we say that G_n is *substitutable by* G' if $in_n = s$ and $Exit_n = T$. The HG $\mathcal{G}[G_n \leftarrow G']$ is

then $\langle G_1, \ldots, G_{n-1}, G' \rangle$. We say that a function $f : \mathscr{G} \to \mathscr{G}$ *maintains* P if for every graph $G \in \mathscr{G}$, we have that G is substitutable by $f(G)$ and for every HG $\mathcal{G}$ with bottom subgraph G, we have $P(\mathcal{G}) = P(\mathcal{G}[G \leftarrow f(G)])$. Note that, in particular, if $\mathcal{G}$ contains only one flat subgraph G, then solving P in G can be done by solving P in $f(G)$.

Example 2. Let P be the problem of finding the length of a shortest path from s to some vertex in T in a possibly weighted graph. Then a function f_{tree} that maps G to the tree $\{s\} \times T$ in which the weight of an edge $\langle s, t \rangle$ is the length of the shortest path from s to t maintains P. □

Consider two complexity functions $g_s : \mathbb{N} \to \mathbb{N}$ and $g_t : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, where s and t stand for *size* and *time*, respectively. Note that g_t has two parameters. We say that a problem P is (g_s, g_t)-*compressible* if there is a function $f : \mathscr{G} \to \mathscr{G}$ such that the following hold.

1. f maintains P,
2. For every graph $G = \langle V, s, T, E \rangle$, we have that $|f(G)| \leq g_s(|T|)$, and
3. For every graph $G = \langle V, s, T, E \rangle$, the complexity of calculating $f(G)$ is $g_t(|G|, |T|)$.

We then say that P is (g_s, g_t)-compressible with witness f.

Note that, by Condition 2, the size of $f(G)$ depends only on the number of exit vertices in G. In particular, if G has a constant number of exit vertices, then $f(G)$ is of constant size.

Theorem 1. *Let $\mathcal{G}$ be an HG. If a problem P is (g_s, g_t)-compressible, then we can generate a flat graph G such that $P(\mathcal{G}) = P(G)$, $|G| \leq g_s(|Exit_1|)$, and the complexity of calculating G is $\sum_{i=1}^{n} g_t(|G_i| - |B_i| + \sum_{b \in B_i} g_s(|Exit_{\tau_i(b)}|), |Exit_i|)$.*

Proof. Let $\mathcal{G} = \langle G_1, \ldots, G_n \rangle$, with $G_i = \langle V_i, B_i, in_i, Exit_i, \tau_i, E_i \rangle$. We prove that for every $0 \leq j \leq n-1$, we can generate an HG $\mathcal{G}'_j = \langle G^j_1, G^j_2, \ldots, G^j_{n-j} \rangle$ such that:

1. $P(\mathcal{G}) = P(\mathcal{G}'_j)$,
2. $|G^j_{n-j}| \leq g_s(|Exit_{n-j}|)$, and
3. The complexity of calculating $\mathcal{G}'_j$ is $\sum_{i=n-j}^{n} g_t(|G_i| - |B_i| + \sum_{b \in B_i} g_s(|Exit_{\tau_i(b)}|), |Exit_i|)$.

The theorem then follows by taking $j = n-1$. Indeed, $\mathcal{G}'_{n-1} = \langle G^{n-1}_1 \rangle$ consists of a single subgraph, which must be flat.

Let $f : \mathscr{G} \to \mathscr{G}$ be such that P is (g_s, g_t)-compressible with witness f. The proof proceeds by an induction on j. For $j = 0$, we define $\mathcal{G}'_0 = \mathcal{G}[G_n \leftarrow f(G_n)]$. The claim follows immediately from the fact P is (g_s, g_t)-compressible with witness f.

Assume the claim holds for $j \in \{0, \ldots, k-1\}$. We prove it holds for $j = k$. We define $\mathcal{G}'_k = \mathcal{G}'_{k-1} \downarrow [G^k_{n-k} \leftarrow f(G^k_{n-k})]$. That is, we flatten all boxes that call the bottom subgraph in $\mathcal{G}'_{k-1}$, obtain an HG with $n-k$ subgraphs, and then apply f to the new bottom subgraph, namely on G^k_{n-k}. We prove that all three conditions hold.

1. Since flattening maintains P and so does an application of f, then $P(\mathcal{G}'_k) = P(\mathcal{G}'_{k-1})$. Thus, by the induction hypothesis, $P(\mathcal{G}) = P(\mathcal{G}'_k)$.
2. All subgraphs $G^0_{n-k}, G^1_{n-k}, \ldots, G^k_{n-k}$ have the same set of exit vertices, namely $Exit_{n-k}$. By the definition of (g_s, g_t)-compressibility, $|f(G^k_{n-k})| \leq g_s(|Exit_{n-k}|)$.
3. In particular, note that $|G^k_{n-k}|$ does not depend on the size of internal subgraphs, as f does not change the number of exit vertices. Finally, the calculation of $\mathcal{G}'_k$ involves a calculation of $\mathcal{G}'_j$ for all $j \in \{0, \ldots, k-1\}$. For every $j \in \{0, \ldots, k-1\}$, the calculation of f on G^j_{n-j}, which is required in order to obtain $\mathcal{G}'_j$, is done on a graph of size $|G_{n-j}| - |B_{n-j}| + \sum_{b \in B_{n-j}} g_s(|Exit_{\tau_{n-j}(b)}|)$ with $|Exit_{n-j}|$ exit vertices. Hence, the time complexity of calculating G'_j is $\sum_{i=n-j}^{n} g_t(|G_i| - |B_i| + \sum_{b \in B_i} g_s(|Exit_{\tau_i(b)}|), |Exit_i|)$.

□

We focus on linear, polynomial, and exponential functions. For classes $\gamma_G, \gamma_T \in \{\text{LIN}, \text{POLY}, \text{EXP}\}$, we say that a problem P is (γ_G, γ_T)-compressible if P is (g_s, g_t)-compressible for g_s whose complexity is γ_T in its parameter, and g_t that is γ_G in its first parameter and γ_T in its second parameter. For example, if $g_s(y) = 2^y$ and $g_t(x, y) = x + 2^y$, then P is (LIN, EXP)-compressible.

Theorem 2. *Let P be a (γ_G, γ_T)-compressible problem on graphs.*

1. *If γ_G and γ_T are* LIN *(respectively,* POLY*), then the complexity of solving P in HGs is linearly- (polynomially-) reducible to the problem of solving P in flat graphs.*
2. *If γ_G is* LIN *(respectively,* POLY*), then the complexity of solving P in CE-HGs is linear (polynomial).*

3.1 Applications

Shortest Path. The problem P from Example 2 is (POLY, POLY)-compressible. Indeed, calculating a tree $\{s\} \times T$ with the length of the shortest path from s to t as the weight of every edge $\langle s, t \rangle$ can be done in polynomial time, and the size of such a tree is clearly linear in $|T|$. Theorem 2 thus implies that the shortest-path problem for HGs can be solved in polynomial time.

Eulerian Cycle. In the full version we present a polynomial function f that maintains the Eulerian cycle problem. Intuitively, f preserves both the parity of the degree of every interface vertex, and the connectivity of the graph. Technically, this is done by maintaining the interface vertices of the bottom subgraph and adding internal vertices for each maximal set of connected interface vertices. The new vertices are connected in a way that preserves connectivity and parity of degrees (parity of both in- and out-degrees, in the case of a connected graph).

Alternating Reachability. A directed AND-OR graph $\langle V, E\rangle$ is a graph whose vertices are partitioned into AND and OR vertices. Thus, $V = V_{and} \cup V_{or}$, with $V_{and} \cap V_{or} = \emptyset$. We say that a set T of vertices is alternating-reachable from a vertex u, denoted $ar(u, T)$, if $u \in T$, $u \in V_{or}$ and T is alternating-reachable from some successor of u, or $u \in V_{and}$ and T is alternating-reachable from all the successors of u. The AR problem is to decide, given G, u, and T, whether $ar(u, T)$. The problem is PTIME-complete [12] for flat graph and PSPACE-complete for HGs [16]. We prove that AR is (LIN, EXP)-compressible, implying it is PTIME-complete for CE-HGs.

We describe a function f that maintains P. Consider an AND-OR graph $G = \langle V, s, T, E\rangle$, with a partition $V_{and} \cup V_{or}$. Let $T_1, \ldots, T_m$ be the subsets of T that are alternating-reachable from s. We define $f(G) = \langle \{s, u_1, \ldots, u_m\} \cup T, s, T, E'\rangle$, where the OR-vertices are $\{s\} \cup (T \cap V_{or})$, and the AND-vertices are $\{u_1, \ldots, u_m\} \cup (T \cap V_{and})$. The edge relation E' consists of edges $\{s\} \times \{u_1, \ldots, u_m\}$ and $\{u_j\} \times T_j$ for every $1 \leq j \leq m$. That is, there are edges from s to every vertex u_j, and from u_j to every vertex in T_j, where $1 \leq j \leq m$. It is easy to see that calculating $f(G)$ is polynomial in $|G|$ and exponential in $|T|$, and its size is exponential in $|T|$.

Maximal Flow. A flow network is a weighted directed graph $G = \langle V, s, \{t\}, E, c\rangle$, where the weight function $c : E \to \mathbb{N}$ maps each edge to its capacity, namely the maximum amount of flow that can travel through it, the source edge s has no incoming edges, and the target vertex t has no outgoing edges. A *flow* is a mapping $f : E \to \mathbb{R}^+$ that satisfies the following two constraints: (1) For every edge $(u, v) \in E$, it holds that $f(u, v) \leq c(u, v)$, and (2) For every $v \in V \setminus \{s, t\}$, it holds that $\sum_{u:(u,v)\in E} f(u, v) = \sum_{u:(v,u)\in E} f(v, u)$. The *value of flow* is defined by $|f| = \sum_{v:(s,v)\in E} f(s, v)$. The problem of finding a maximal flow (MF, for short) is to find, given a flow network G, the maximal value of a flow for it. For flat graphs, the problem can be solved in polynomial time (cf., the Ford-Fulkerson algorithm [9]). It is shown in [16] that the problem is PSPACE-complete for hierarchical graphs. Here, we conjecture that the problem is (POLY,EXP)-compressible, implying that it can be solved in PTIME for CE-HGs. We prove that for graphs with at most 3 exit vertices, it is even (POLY,POLY)-compressible, implying it can be solved in PTIME for such graphs. For the general case, the problem is strongly related to the problem of polytop optimization (cf., [20]).

Consider a flow graph with source and several target vertices $G = \langle V, s, T, E, c\rangle$. The *characteristic function* $v : 2^T \to \mathbb{R}^+$ of G maps each a set $U \subseteq T$ of target vertices to the maximal flow that can be routed to U. That is, $v(U)$ is the maximal flow in a network $G' = \langle V \cup \{t'\}, s, \{t'\}, E', c'\rangle$, obtained from G by defining a new single sink vertex t' and adding edges with capacity ∞ from every vertex in U to t'. In particular, $v(T)$ is the maximal flow that can be routed to all target vertices together. Also, for a single vertex $t \in T$, we have that $v(t)$ is the maximal flow that can be routed to t. It is known that the characteristic function is necessary and sufficient in order to represent a flow

network [18]. Therefore, the equivalence of two flow networks can be proved by proving that their characteristic functions are identical.

We show a function f that maintains the MF problem for flow networks with up to three sink vertices: given a flow network $G = \langle V, s, T, E, w\rangle$, where $|T| \leq 3$, the function $f(G)$ constructs in polynomial time a flow network G' of constant size such that every flow in G is a flow in G', and vice versa. It is easy to see that every flow network with one sink t is equivalent to a flow network with two vertices, s and t, connected by an edge with capacity $v(t)$. Formally, if a given flow network $G = \langle V, s, \{t\}, E, w\rangle$ has one exit vertex, then $f(G) = \langle \{s, t\}, s, \{t\}, \langle s, t\rangle, v(t)\rangle$. In Fig. 2, we describe the flow network that f returns for flow networks with two (left) and three (right) sinks. For the one in the right, it is easy to see that the maximal flow in this network is $v(\{t_1, t_2\})$, while the maximal flow to the sinks t_1 and t_2 is $v(t_1)$ and $v(t_2)$, respectively. In addition, f maintains MF also for three sinks. Note that calculating f requires $2^{|T|} - 1$ polynomial computations, where $|T| \leq 3$.

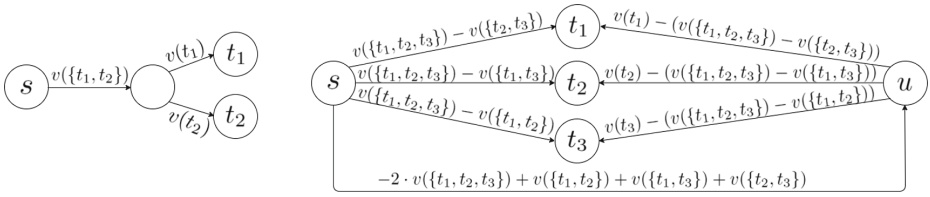

Fig. 2. Compressing flow networks with two and three sinks

Conjecture 1. The problem of maximal flow is (POLY, EXP)-compressible. Thus, by Theorem 2 it can be solved in PTIME for CE-FGs.

3.2 Not All Problems Are Compressible

We conclude this section pointing to a problem for which the exponential penalty cannot be avoided even if the number of exit vertices is constant. The problem of *path of length in a given interval in a tree* (the PI problem, for short) is to decide, given a weighted directed tree $G = \langle V, E, w\rangle$ and an interval $[x, y]$, whether there is a path of length in $[x, y]$ in G. For flat graphs, we can solve the problem by computing the length of the path between every pair of vertices. Since the graph is a directed tree, there is at most one path between every two nodes, and therefore this algorithm is polynomial. A tree HG is an HG all whose subgraphs are trees.

Lemma 1. *The PI problem is NP-complete for CE-HGs.*

4 Nondeterministic Compression of Hierarchical Graphs

In this section we study problems P for which a polynomial verifier is known in the flat setting. We argue that when the number of exit vertices is bounded by a constant, this can be used in order to verify that a suggested compression of the bottom subgraph indeed maintains P. This enables us to show membership in NP for the hierarchical setting for several problems for which membership in NP is known in the flat setting, improving the PSPACE upper bound for the general case (that is, when the number of exit vertices is not a constant).

Consider a decision problem P and a graph with entry and exit vertices G. We say that G is *hopeful with respect to* P if there is an HG in which G is called and $P(\mathcal{G})$ holds. For example, if P is 3-coloring, and G includes a 4-clique, then G is not hopeful. We say that a relation $R : \mathscr{G} \times \mathscr{G}$ *maintains* P if for every graph $G \in \mathscr{G}$ that is hopeful with respect to P there is at least one graph $G' \in \mathscr{G}$ such that $\langle G, G' \rangle \in R$, and for every pair of graphs $\langle G, G' \rangle \in R$, we have that G is substitutable by G' and for every HG $\mathcal{G}$ with bottom subgraph G, we have $P(\mathcal{G}) = P(\mathcal{G}[G \leftarrow G'])$.

Consider three complexity functions $g_s : \mathbb{N} \to \mathbb{N}$, $g_t : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and $g_w : \mathbb{N} \to \mathbb{N}$, where s, t and w stand for *size*, *time* and *witness*, respectively. We say that a problem P is *nondeterministically-*(g_s, g_t, g_w)*-compressible* if there is a relation $R : \mathscr{G} \times \mathscr{G}$ such that the following hold.

1. R maintains P,
2. For every graph $G = \langle V, s, T, E \rangle$ and pair $\langle G, G' \rangle \in R$, we have that $|G'| \leq g_s(|T|)$,
3. There is a verifier $\mathcal{V}$ that runs in time $g_t(|G|, |T|)$ such that whenever $\langle G, G' \rangle \in R$, there is a witness w such that $\mathcal{V}$ returns "yes" on $(\langle G, G' \rangle, w)$, and
4. For every graph $G = \langle V, s, T, E \rangle$, we have $|\{G' : \langle G, G' \rangle \in R\}| \leq g_w(|T|)$.

We then say that P is nondeterministically-(g_s, g_t, g_w)-compressible with witness R.

Theorem 3. *Let $\mathcal{G}$ be an HG. If a problem P is nondeterministically-(g_s, g_t, g_w)-compressible and $P(\mathcal{G})$ holds, then there is a flat graph G such that $|G| \leq g_s(|Exit_1|)$, and there is a verifier $\mathcal{V}$ that runs in time $\sum_{i=1}^{n} g_w(|Exit_i|) \cdot (g_t(|G_i| - |B_i| + \sum_{b \in B_i} g_s(|Exit_{\tau_i(b)}|), |Exit_i|))$ such that if $P(\mathcal{G}) = P(G)$, then there is a witness w such that $\mathcal{V}$ returns "yes" on (G, w).*

Theorem 4. *If a problem P is nondeterministically-(g_s, g_t, g_w)-compressible for g_t polynomial in its first parameter, then P is in NP for CE-HGs.*

4.1 Applications

In this section we describe NP-complete problems that stay in NP for CE-HGs. We rely on Theorem 4, showing that the problems are nondeterministically-(g_s, g_t, g_w)-compressible for g_t that is polynomial in its first parameter in $|G|$.

k-coloring. A valid k-coloring for a graph is a labeling of the vertices of the graph by k colors so that adjacent vertices are mapped to different colors. We define a relation $R : \mathcal{G} \times \mathcal{G}$ such that for every graph $G = \langle V, s, T, E\rangle$, we have that $\langle G, G'\rangle \in R$ iff $G' = \langle \{s, v_1, v_2, \ldots, v_k\} \cup T, s, T, E'\rangle$, there is exactly one valid coloring of G' with k colors, and there is a valid coloring of G that agrees with the coloring of the interface vertices in G'. Note that if there is a valid coloring of the interface vertices in G, then the edge relation E' can force it on the interface vertices in G' using a "k-colors plate" clique $\{v_1, v_2, \ldots, v_k\}$. Namely, the set E' includes an edge between every pair of vertices in $\{v_1, v_2, \ldots, v_k\}$, and in addition it includes edges from every vertex $v \in \{s\} \cup T$ to the colors in $\{v_1, v_2, \ldots, v_k\}$ that v is not colored in. It is easy to see that R maintains P. Note that for every graph $G = \langle V, s, T, E\rangle$ and pair $\langle G, G'\rangle \in R$, $|G'|$ is clearly of size linear in $|T|$, and verifing that $\langle G, G'\rangle \in R$ can be done in polynomial time, given a witness that shows a coloring of G that agrees with G' on the coloring of the interface vertices. Finally, for every graph G, the number of graphs G' such that $\langle G, G'\rangle \in R$ is bounded by $k^{|T|}$, which is constant for CE-HGs.

Independent Set. The independent set problem is to decide, given a graph $G = \langle V, E\rangle$ and number $k \geq 0$, whether G contains an independent set $S \subseteq V$ of size at least k, where S is independent if for all two vertices v and v' in S, we have $(v, v') \notin E$. Note that the input to the problem contains a parameter k. We extend our definition of maintenance to also account for the parameter. Thus, the relation $R : (\mathcal{G} \times \mathbb{N}) \times (\mathcal{G} \times \mathbb{N})$ *maintains* P if for every graph $G \in \mathcal{G}$ and parameter $k \in \mathbb{N}$, such that $\langle G, k\rangle$ is hopeful with respect to P, there is at least one graph $G' \in \mathcal{G}$ and parameter $k' \in \mathbb{N}$, such that $\langle (G, k), (G', k')\rangle \in R$. In addition, we allow R to refer to labeled versions of G and G'. Formally, we rephrase the independent set problem as follows: given $G = \langle V, E\rangle$ and k, decide whether there is a labeling function $l : V \to \{0, 1\}$ such that $l^{-1}(1)$ forms an independent set of size at least k. Now, whenever we compress a graph G, we mark the interface vertices of G' by 0's and 1's, serving as identifiers as to whether these vertices participate in the witness independent set of G. Thus, $\langle (G, k), (G', k - k')\rangle \in R$ iff $G' = \langle \{s\} \cup T, s, T, \emptyset, l\rangle$, where l is such that there is an independent set S in G of size $k' + |(\{s\} \cup T) \cap l^{-1}(1)|$ such that for all $v \in \{s\} \cup T$, we have that $v \in S$ iff $l(v) = 1$. Note that l uniquely defines k', thus for each $l : \{s\} \cup T \to \{0, 1\}$, there is at most one graph G' such that $\langle (G, k), (G', k - k')\rangle \in R$.

Now, when G is replaced by G', the search for a labeling function in $\mathcal{G}$ that serves as a characteristic function of the independent set takes the labels of G' into account. Thus, edges that lead to an interface vertex of G' that is labeled by 1 cannot be labeled by 1. Since there are at most $2^{|T|+1}$ possible labeling functions, all the conditions on R are satisfied.

Hamilton Path. A Hamiltonian path in the graph is a path that traverses all the vertices of the path, each vertex exactly once. For directed graphs, the hierarchical setting is less challenging, as a subgraph may be entered only once.

We consider here also the more challenging undirected case, where a subgraph may be entered via all its interface vertices. There, a Hamiltonian path in an HG may enter a subgraph more than once, and it may visit an interface vertex of a box without entering or leaving it. For an HG $\mathcal{G}$ with a box b that is substituted by G, restricting a Hamiltonian path to b yields a set of paths in G, where every vertex in G appears exactly in one path. This set induces a partition of the interface vertices of G, where the vertices in every subset in the partition are ordered. For a Hamiltonian path with sub-paths $p_1, \ldots, p_m$ in G, we denote this partition as *the restriction of* $p_1, \ldots, p_m$ *to the interface vertices of* G. We define a relation $R : \mathscr{G} \times \mathscr{G}$ such that for every graph $G = \langle V, s, T, E \rangle$, we have that $\langle G, G' \rangle \in R$ iff $G' = \langle \{s\} \cup T, s, T, E' \rangle$, and E' is such that there are two sets of paths $\{p_1, \ldots, p_m\}$ in G and $\{p'_1, \ldots, p'_m\}$ in G' with the same restriction to the interface vertices $\{s\} \cup T$. It is easy to see that this relation maintains the problem of Hamiltonian path both for directed and undirected graphs and for every graph, there is a bounded number of graphs that are in relation R with it. In addition, for every graph $G = \langle V, s, T, E \rangle$ and pair $\langle G, G' \rangle \in R$, $|G'|$ is clearly of size linear in $|T|$. Finally, given a witness that shows set of paths in G that are restricted to the same partition as E' in G', verifing that $\langle G, G' \rangle \in R$ can be easily done in polynomial time.

References

1. Alur, R., Kannan, S., Yannakakis, M.: Communicating hierarchical state machines. In: Wiedermann, J., Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 169–178. Springer, Heidelberg (1999). doi:10.1007/3-540-48523-6_14
2. Alur, R., Yannakakis, M.: Model checking of hierarchical state machines. ACM TOPLAS **23**(3), 273–303 (2001)
3. Aminof, B., Kupferman, O., Murano, A.: Improved model checking of hierarchical systems. Inf. Comput. **210**, 68–86 (2012)
4. Barrett, C., Jacob, R., Marathe, M.: Formal-language-constrained path problems. SIAM J. Comput. **30**(3), 809–837 (2000)
5. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (1999)
6. de Roever, W.-P.: The need for compositional proof systems: a survey. In: de Roever, W.-P., Langmaack, H., Pnueli, A. (eds.) COMPOS 1997. LNCS, vol. 1536, pp. 1–22. Springer, Heidelberg (1998). doi:10.1007/3-540-49213-5_1
7. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer, Heidelberg (2013)
8. Drusinsky, D., Harel, D.: On the power of bounded concurrency I: finite automata. J. ACM **41**(3), 517–539 (1994)
9. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. Can. J. Math. **8**(3), 399–404 (1956)
10. Galperin, H., Wigderson, A.: Succinct representations of graphs. Inf. Control **56**(3), 183–198 (1983)
11. Harel, D., Kupferman, O., Vardi, M.Y.: On the complexity of verifying concurrent transition systems. Inf. Comput. **173**, 1–19 (2002)
12. Immerman, N.: Length of predicate calculus formulas as a new complexity measure. In: Proceedings of 20th FOCS, pp. 337–347 (1979)

13. Kupferman, O., Tamir, T.: Hierarchical network formation games. In: Legay, A., Margaria, T. (eds.) TACAS 2017. LNCS, vol. 10205, pp. 229–246. Springer, Heidelberg (2017). doi:10.1007/978-3-662-54577-5_13
14. Kupferman, O., Vardi, M.Y., Wolper, P.: An automata-theoretic approach to branching-time model checking. J. ACM **47**(2), 312–360 (2000)
15. Lengauer, T.: The complexity of compacting hierarchically specified layouts of integrated circuits. In: Proceedings of 23rd FOCS, pp. 358–368 (1982)
16. Lengauer, T., Wagner, K.W.: The correlation between the complexities of the non-hierarchical and hierarchical versions of graph problems. JCSS **44**, 63–93 (1990)
17. Lengauer, T., Wanke, E.: Efficient solutions of connectivity problems on hierarchically defined graphs. SIAM J. Comput. **17**(6), 1063–1081 (1988)
18. Megiddo, N.: Optimal flows in networks with multiple sources and sinks. Math. Program. **7**(1), 97–107 (1974)
19. Mendelzon, A.O., Wood, P.T.: Finding regular simple paths in graph databases. SIAM J. Comput. **24**(6), 1235–1258 (1995)
20. Rothvoß, T.: The matching polytope has exponential extension complexity. In: Proceedings of 46th STOC, pp. 263–272 (2014)