Compositional Algorithms on Compositional Data: Deciding Sheaves on Presheaves

Benjamin Merlin Bumpus

Algorithmicists are well-aware that fast dynamic programming algorithms are very often the correct choice when computing on compositional (or even recursive) *graphs*. Here we initiate the study of how to generalize this folklore intuition to mathematical structures writ large. We achieve this horizontal generality by adopting a categorial perspective which allows us to show that: (1) *structured decompositions* (a recent, abstract generalization of many graph decompositions) define Grothendieck topologies on categories of data (adhesive categories) and that (2) any computational problem which can be represented as a *sheaf* with respect to these topologies can be decided in linear time on classes of inputs which admit decompositions of bounded width and whose decomposition shapes have bounded feedback vertex number. This immediately leads to algorithms on objects of any C-set category; these include – to name but a few examples – structures such as: symmetric graphs, directed graphs, directed multigraphs, hypergraphs, directed hypergraphs, databases, simplicial complexes, circular port graphs and half-edge graphs. Finally we pair our theoretical results with concrete implementations of our main algorithmic contribution in the *AlgebraicJulia* ecosystem. The paper-length version of this extended abstract – which is *joint work Ernst Althaus, James Fairbanks and Daniel Rosiak* – is available at https://arxiv.org/abs/2302.05575.

1 PHILOSOPHY

Among the many different incarnations of compositionality in mathematics, the following three will be the main characters of this story: (1) the *structural* compositionality arising in graph theory in the form of *graph decompositions* whereby one decomposes graphs into smaller and simpler constituent parts [3–8], (2) the *representational* compositionalityarising in the form of *sheaves* in algebraic topology (and elsewhere) and (3) the *algorithmic* compositionality embodied by the intricate *dynamic programming* routines found in parameterized complexity theory [3, 4, 6–8]. Our main contribution (Theorem 4.2) is an algorithmic meta-theorem obtained by amalgamating these three perspectives.

2 DECISION PROBLEMS

Computational problems are assignments of data – thought of as *solution spaces* – to some class of input objects. We think of them as functors $\mathcal{F}\colon \mathsf{C} \to \mathsf{Sol}$ taking objects of some category C to objects of some appropriately chosen category Sol of solution spaces. Rather than computing the entire solution space, we often have to settle for more approximate representations of the problem – in the form of decision/optimization/enumeration problems. When does a given computational problem admit a compositional structure that is well-behaved with respect to decompositions of the input data? One of our motivations was the observation that sheaves may be applicable to this situation. Here we will focus on *decision problems*: for a given computational problem $\mathcal{F}$ we define the **$\mathcal{F}$-decision problem** as the composite $\mathsf{C} \xrightarrow{\mathcal{F}} \mathsf{Sol} \xrightarrow{\mathsf{dec}} \mathbf{2}$ where dec is a functor into $\mathbf{2}$ mapping solutions spaces to either $\perp$ or $\top$ depending on whether they witness yes- or no-instances to $\mathcal{F}$. For example consider the $\textsc{GraphColoring}_n$ problem[1]; it can easily be encoded as the representable functor $\mathsf{SimpFinGr}(-, K_n)\colon \mathsf{SimpFinGr}^{op} \to \mathsf{FinSet}$ on the category of finite simple graphs. One turns this into decision problems by taking $\mathsf{dec}\colon \mathsf{FinSet} \to \mathbf{2}$ to be the functor which takes any set to $\perp$ if and only if it is empty. By passing to other categories of graphs (for instance that of graphs and their *monomorphisms*) one can also easily encode other decision problems such as VertexCover and OddCycleTransversal.

3 COMPOSITIONAL DATA

Parameterized complexity [4] is a two-dimensional framework for complexity theory whose main insight is that one should not analyze running times only in terms of the total input size, but also in terms of other *parameters* of the inputs (such as measures of *compositional structure* [4]). Here we represent compositional structure via *diagrams*: we think of an object $c \in \mathsf{C}$ obtained as the colimit of a diagram $d\colon \mathsf{J} \to \mathsf{C}$ as being *decomposed* by d into smaller constituent pieces. In particular we work with a special class of diagrams (suited for algorithmic manipulation) called **structured decompositions** [2]. Roughly they consist of a collection of objects in a category and a collection of *spans* which relate these objects (just like the edges in a graph relate its vertices).

Definition 3.1. Let G be a graph viewed as a C-set. Fixing a base category K, we define a **K-valued structured decomposition of shape G** as a diagram of the form $d\colon \int G \to \mathsf{K}$ whose arrows are all *monic* in K. (Note

[1] It asks to determine whether a given input graph G admits a proper coloring with at most n colors.

that, the Grothendieck construction applied to a graph G yields a category whose arrows form spans from each edge of G to its endpoints.) Structured decompositions assemble into a subcategory $\mathfrak{D}_{\mathrm{m}}\,\mathsf{K}$ of the category of diagrams in K.

4 DECISION PROBLEMS AS SHEAVES

One of our main contributions is to show that structured decompositions yield Grothendieck topologies [2] on *adhesive* categories which are *adhesively cocomplete*[3].

Theorem 4.1. *Let* C *be a small adhesively cocomplete category. The following is a contravariant functor by pullback.*

$$\mathsf{Dcmp}\colon \mathsf{C}^{op} \to \mathbf{Set} \tag{1}$$

$$\mathsf{Dcmp}\colon c \mapsto \{\Lambda_d \mid \operatorname{colim} d = c \text{ and } d \in \mathfrak{D}_{\mathrm{m}}\,\mathsf{C}\}$$

$$\bigcup \{\{f\} \mid f\colon a \xrightarrow{\cong} c \text{ an iso }\}.$$

This functor makes the pair $(\mathsf{C}, \mathsf{Dcmp})$ *a site.*

This result allows us to consider those decision problems which display compositional structure compatible with that highlighted by structured decompositions; namely any problem $\mathcal{F}\colon \mathsf{C}^{op} \to \mathsf{Set}$ which is a *sheaf* with respect to the decomposition topology of Theorem 4.1.

Roughly, our main algorithmic result shows that any such problem can be solved in time that grows linearly in the size of the decomposition and exponentially in in terms of the internal complexity of the constituent parts and the feedback vertex number[4] of the shape of the decomposition.

Theorem 4.2. *Let G be a finite, irreflexive, directed graph without antiparallel edges and at most one edge for each pair of vertices. Let* C *be a small adhesively cocomplete category, let* $\mathcal{F}\colon \mathsf{C}^{op} \to \mathsf{FinSet}$ *be a presheaf. If* $\mathcal{F}$ *is a sheaf on the site* $(\mathsf{C}, \mathsf{Dcmp})$ *and if we are given an algorithm* $\mathcal{A}_{\mathcal{F}}$ *which computes* $\mathcal{F}$ *on any object c in time $\alpha(c)$, then there is an algorithm which, given any* C*-valued structured decomposition* $d\colon \int G \to \mathsf{C}$ *of an object* $c \in \mathsf{C}$ *and a feedback vertex set S of G, computes* $\mathsf{dec}\,\mathcal{F}\,c$ *in time*

$$(\max_{x \in VG} \alpha(dx) + \kappa^{|S|}\kappa^2)|EG|$$

where $\kappa = \max_{x \in VG} |\mathcal{F}\,dx|$.

We note (just as Bodlaender and Fomin [1] already did) that it is not the width of the decompositions of the inputs that matters; instead it is the *width of the decompositions of the solutions spaces* (whose constituent parts are sets) that is key to the algorithmic bounds. Furthermore, we note we are treating the algorithm $\mathcal{A}_{|}mathcalF$ as if given by an oracle. We state our result in terms of a running time bound on $\mathcal{A}_{|}mathcalF$ to make explicit how, in practice, this computation will impact the overall running time.

4.1 Sketching Theorem 4.2

Notice that, if C has colimits, and is adhesive, then, since sheaves for the Decmp topology preserve the corresponding colimits (sending them to limits of sets) [9], the following diagram will always commute [2].

$$\begin{tikzcd} \mathsf{C} \arrow[r, "\mathcal{F}"] & \mathsf{FinSet}^{op} \arrow[r, "\mathsf{dec}^{op}"] & 2^{op} \\ \mathfrak{D}_{\mathrm{m}}\,\mathsf{C} \arrow[u, "\mathsf{colim}"] \arrow[r, "\mathfrak{D}_{\mathrm{m}}\,\mathcal{F}"'] \arrow[ur, phantom, "comm."] & \mathfrak{D}_{\mathrm{m}}\,\mathsf{FinSet}^{op} \arrow[u, "\mathsf{colim}"'] & \end{tikzcd} \tag{2}$$

Unpacking the diagram, the blue path corresponds to forgetting the compositional structure and then solving the problem on the entire input. On the other hand the red path corresponds to a compositional algorithm for deciding sheaves: one first evaluates $\mathcal{F}$ on the constituent parts of the decomposition and then joins[5] these solutions together to find a solution on the whole.

Unfortunately what we just described is still very inefficient since, for any input c and no matter which path we take in the diagram, we always end up computing all of $\mathcal{F}(c)$: this is very large in general (think of coloring sheaf mentioned above). One might hope to overcome this difficulty by lifting[6] dec to a functor from FinSet^{op}-valued structured decompositions to 2^{op}-valued structured decompositions as is shown in the following diagram.

$$\begin{tikzcd} \mathsf{C} \arrow[r, "\mathcal{F}"] & \mathsf{FinSet}^{op} \arrow[r, "\mathsf{dec}^{op}"] & 2^{op} \\ \mathfrak{D}_{\mathrm{m}}\,\mathsf{C} \arrow[u, "\mathsf{colim}"] \arrow[r, "\mathfrak{D}_{\mathrm{m}}\,\mathcal{F}"'] \arrow[ur, phantom, "comm."] & \mathfrak{D}_{\mathrm{m}}\,\mathsf{FinSet}^{op} \arrow[u, "\mathsf{colim}"'] \arrow[r, "\mathfrak{D}_{\mathrm{m}}\,\mathsf{dec}^{op}"'] & \mathfrak{D}_{\mathrm{m}}\,2^{op} \arrow[u, "\mathsf{colim}=\wedge"'] \end{tikzcd}$$

However, this too is to no avail: the right-hand square of the above diagram does *not* commute in general. The main ingredient in proving our algorithmic result is to show that there is an *efficiently computable* endofunctor $\mathcal{A}\colon \mathfrak{D}_{\mathrm{m}}\,\mathsf{Set}^{op} \to \mathfrak{D}_{\mathrm{m}}\,\mathsf{Set}^{op}$ making the following diagram commute.

---

[2]For the reader concerned with size issues, observe that: (1) given categories C and D, the functor category $[C, D]$ is small whenever C and D also are; and (2) since we are concerned with diagrams whose domains have finitely many objects and morphisms, one has that the collection of diagrams which yield a given object as a colimit is indeed a set.

[3]i.e. they have colimits of diagrams of monomorphisms.

[4]A **feedback vertex set** in a graph G is a vertex subset $S \subseteq V(G)$ of G whose removal from G yields an acyclic graph.

[5]Note that the colimit functor $\mathsf{colim}\colon \mathfrak{D}_{\mathrm{m}}\,\mathsf{FinSet}^{op} \to \mathsf{FinSet}^{op}$ – which is in red in Diagram 2 – takes decompositions to their *limit in* FinSet (since colimits in FinSet^{op} are limits in FinSet); we invite the reader to keep this in mind throughout.

[6]The construction of categories of structured decompositions is functorial [2], this is inherited from the analogous statement for categories of diagrams.

$$\begin{tikzcd}
\mathsf{C} \arrow[r, "\mathcal{F}"] & \mathsf{FinSet}^{op} \arrow[rr, "\mathrm{dec}^{op}"] & & 2^{op} \\
\mathfrak{D}_{\mathrm{m}}\, \mathsf{C} \arrow[u, "\mathsf{colim}"] \arrow[r, "\mathfrak{D}_{\mathrm{m}}\, \mathcal{F}"] \arrow[ur, phantom, "\textit{comm.}"] & \mathfrak{D}_{\mathrm{m}}\, \mathsf{FinSet}^{op} \arrow[u, "\mathsf{colim}"'] \arrow[r, "\mathcal{A}"] \arrow[urr, phantom, "\textit{comm.}"] & \mathfrak{D}_{\mathrm{m}}\, \mathsf{FinSet}^{op} \arrow[r, "\mathfrak{D}_{\mathrm{m}}\, \mathrm{dec}^{op}"] & \mathfrak{D}_{\mathrm{m}}\, 2^{op} \arrow[u, "\wedge"']
\end{tikzcd}$$

REFERENCES

[1] H. L Bodlaender and F. V. Fomin. Tree decompositions with small cost. In *Algorithm Theory—SWAT 2002: 8th Scandinavian Workshop on Algorithm Theory Turku, Finland, July 3–5, 2002 Proceedings 8*, pages 378–387. Springer, 2002.

[2] B. M. Bumpus, Z. A. Kocsis, and J. E. Master. Structured Decompositions: Structural and Algorithmic Compositionality. *arXiv preprint arXiv:2207.06091*, 2022. URL: https://doi.org/10.48550/arXiv.2207.06091.

[3] B. Courcelle and J. Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press, 2012. doi:https://doi.org/10.1017/CBO9780511977619.

[4] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized algorithms*. Springer, 2015. ISBN:978-3-319-21275-3. doi:https://doi.org/10.1007/978-3-319-21275-3.

[5] R. Diestel. *Graph theory*. Springer, 2010. ISBN:9783642142789.

[6] R. G. Downey and M. R. Fellows. *Fundamentals of parameterized complexity*, volume 4. Springer, 2013. URL: https://doi.org/10.1007/978-1-4471-5559-1.

[7] J. Flum and M. Grohe. Parameterized complexity theory. 2006. *Texts Theoret. Comput. Sci. EATCS Ser*, 2006. ISBN:978-3-540-29952-3. doi:https://doi.org/10.1007/3-540-29953-X.

[8] M. Grohe. Descriptive complexity, canonisation, and definable graph structure theory, cambridge university press, cambridge, 2017, x + 544 pp. *The Bulletin of Symbolic Logic*, 23(4):493–494, 2017. ISBN:1079-8986.

[9] D. Rosiak. *Sheaf Theory through Examples*. The MIT Press, 10 2022. URL: 10.7551/mitpress/12581.001.0001.