Binary Commutative Polymorphisms of Core Triads

Michael Wernthaler

January 12, 2021

Contents

1 TODO Abstract

It has been known for a while that for a given graph H the complexity of CSP(H) also known as the H-colouring problem only depends on the set of polymorphisms of H. It follows from the results of Bulatov [4] and of Zhuk [6] from 2017 that H-colouring problem is in P if H has a so-called (4-ary) Siggers polymorphism. In this paper we focus on the case where H is a so-called *triad*, i.e., an orientation of a tree which has a single vertex of degree 3 and otherwise only vertices of degree 2 and 1. We describe an efficient algorithm that checks the existence of siggers polymorphisms for triads up to a certain number/size of vertices/armlength?.

2 TODO Introduction

Let $H = (V, E)$ be a finite directed graph. The *H-colouring problem* (also called the *constraint satisfaction problem for H*) is the problem of deciding for a given finite graph G whether there exists a homomorphism from G to H. Note that if $H = K_k$, the clique with k vertices, then the H-/colouring problem equals the famous k-colouring problem, which is NP-hard for $k \geq 3$ and which can be solved in polynomial time if $k \leq 2$.

It has been known for a while that the complexity of the H-colouring problem only depends on the set of *polymorphisms*. It follows from results of Bulatov [4] and of Zhuk [6] from 2017 that the H-colouring problem is in P of H has a so-called *(4-ary) Siggers* polymorphism, i.e., an operation $s : V^4 \longrightarrow V$ which satisfies for all $a, e, r \in V$

$$s(a, r, e, a) = s(r, a, r, e)$$

Before these results, the complexity of CSP(H) was open even if H is an orientation of a tree. It is not obvious at all how an orientation of a tree looks like if it has a Siggers polymorphism. In fact, this question is already open if H is a *triad*, i.e., an orientation of a tree which has a single vertex of degree 3 and otherwise only vertices of degree 2 and 1. Jakob Bulin claims that the following triad with 22 vertices has no Siggers polymorphism.

$$01001111, 0110000, 101000$$

Here, 0 stands for forward edge, 1 stands for backward edge, and the three words stand for the three paths that leave the vertex of degree 3 of the triad. He also claims that all smaller triads do have a Siggers polymorphism, and conjectures that an orientation of a tree has a Siggers polymorphism if and only if it has a binary polymorphism f satisfying $f(u, y) = f(y, x)$ for all $x, y \in V$. Jakub Bulin conjectures that in this case the Path-Consistency algorithm can solve the H-colouring problem.

3 DONE Task 1: Lemma

To ensure that our algorithm runs correctly we are first going to prove the following lemma

Lemma 1. *Let $\mathbb{T}$ be a finite tree. The following are equivalent*

1. *$\mathbb{T}$ is a core*
2. *$End(\mathbb{T}) = \{id\}$*
3. *$AC_{\mathbb{T}}(\mathbb{T})$ terminates with $L(v) = v$ for all vertices v of $\mathbb{T}$*

3.1 Proof

$1 \implies 2$: Let $\mathbb{T}$ be a core. Let's assume there is another homomorphism $f \in End(\mathbb{T})$ with $f \neq id$. Knowing that $\mathbb{T}$ is a tree we conclude there must be a leaf u on which f is not the identity. We consider $p = u_0 u_1 ... u_l$ to be the unique path from u to $f(u)$, which maps to the unique path from $f(u)$ to $f(f(u))$.

Now let $\mathbb{T} = v(\xi_1, ..., \xi_k)$ with $\xi_a \to \xi_b$ for at least one pair ξ_a, ξ_b, where ξ_a contains u and ξ_b contains $f(u)$. We then construct a nonbijective endomorphism h of $\mathbb{T}$ by taking f on ξ_a. For every other component we define h as id. But then $\mathbb{T}$ can't be a core, which means our assumption was wrong and $End(\mathbb{T})$ cannot contain such a f, but only id.

$2. \implies 1$: If $End(\mathbb{T}) = id$, then the only homomorphism $h : \mathbb{T} \to \mathbb{T}$ is id, which is an automorphism. Hence $\mathbb{T}$ must be a core.

$2. \implies 3$: Suppose that $End(\mathbb{T}) = \{id\}$. Now let's assume, that $AC_{\mathbb{T}}(\mathbb{T})$ derives $L(v)$ such that it contains another vertex $u \neq v$.

By choosing a vertex from the list of each node we construct a sequence $f_0, ..., f_n$ for $n = |V(\mathbb{T})|$, where f_i is a homomorphism from the subgraph of $\mathbb{T}$ induced by the vertices at distance at most i to v in $\mathbb{T}$, and f_{i+1} is an extension of f_i for all $1 \leq i \leq n$. The mapping f_0 maps v to u. Suppose inductively, that we have already defined f_i. Let w be a vertex at distance $i + 1$ from v in $\mathbb{T}$. Since $\mathbb{T}$ is an orientation of a tree, there is a unique $w' \in V(\mathbb{T})$ of distance i from v in $\mathbb{T}$ such that $(w, w') \in E(\mathbb{T})$ or $(w', w) \in E(\mathbb{T})$. Note that

$x = f_i(w')$ is already defined. In case that $(w', w) \in E(\mathbb{T})$, there must be a vertex y in $L(w)$ such that $(x, y) \in E(\mathbb{T})$, since otherwise the arc-consistency procedure would have removed x from $L(w')$. We then set $f_{i+1}(w) = y$. In case that $(w, w') \in E(\mathbb{T})$ we can proceed analogously. By construction, the mapping f_n is an endomorphism of $\mathbb{T}$ that maps v to u.

But we know that $End(\mathbb{T}) = \{id\}$, so $L(v)$ can not contain such a vertex u, but only v. Hence $L(v) = \{v\}$.

3. $\implies$ 2: It's obvious, that always $\{id\} \subseteq End(\mathbb{T})$. Since $AC_{\mathbb{T}}(\mathbb{T})$ derived $L(v) = v$ for all vertices v of $\mathbb{T}$ we know there can't be another homomorphism h for which $h(v) \neq v$, hence $End(\mathbb{T}) = \{id\}$.

3.2 Notes

4 DONE Task 2: Arc-Consistency Procedure

Implement the arc-consistency procedure such that your algorithm runs in linear time in the size of the input.

Algorithm 1: $AC_{\mathbb{T}}$ ($\mathbb{T}$ is a triad)
1 Input: digraph $\mathbb{G}$, initial lists $L : G \mapsto P(T)$ Output: Is there a homomorphism $h : \mathbb{G} \mapsto \mathbb{T}$ such that $h(v) \in L(v)$ for all $v \in G$

4.1 Notes

- Can we optimize AC for paths?
- Done by implementing AC-3 for graphs

4.2 Benchmarks

Algorithm 2: Algorithm for finding core triads

```
Input: An unsigned integer m
Output: A list of all core triads whose arms each have a length ≤ m

// Finding a list of RCAs
armlist ⟵ [ ];
foreach arm p with length(p) ≤ m do
    if ACR_p(p) didn't derive L(v) ≠ v for any vertex v then
        put p in armlist

// Assembling the RCAs to core triads
triadlist ⟵ [ ];
foreach {p1, p2} in armlist do
    if ACR_{p1p2}(p1p2) derived L(v) ≠ v for some vertex v then
        Drop the pair and cache the two indices;

foreach triad T = {p1, p2, p3} do
    if T contains a cached index pair then
        Drop T and continue;
    if AC_T(T) didn't derive L(v) ≠ v for some vertex v then
        Put T in triadlist;

return triadlist
```

5 DONE Task 3: Core Triads

Write an algorithm that enumerates all core triads up to a fixed path-length.

5.1 Algorithm

Algorithm 2 displays the pseudo-code of the entire core triad generation.

5.2 Notes

5.2.1 Observations

- Let n be the maximal arm length
- Then the number of possible paths is $p = \sum_{i=1}^{n} 2^i$ and there are p^3 possible triads.
- To reduce the number of cases to look at we consider only triads that

are cores, i.e., not homomorphically equivalent to smaller triads. Thus we pose/have to answer? the following question

- **Question 1.** *When is a triad homomorphically equivalent to a smaller triad?*
- A method to answer this question has already been presented in Lemma 1: We simply run $AC_{\mathbb{T}}(\mathbb{T})$ and see if it derives $L(v) = \{v\}$ for every vertex v
- Not efficient! and our algorithm will build up triads from arms
- Obvious case: A triad with two identical arms is obviously not a core triad
- We introduce some further definitions:
- We consider a *partial triad* θ to be a triad of the form $(p_1 p_2 p_3)$ where at least one $p_i = \varepsilon$. If $p_j \neq \varepsilon$ for only one j then we call θ an *arm*. Each partial triad can be completed to form a triad by adding arms to it.
- We cannot be certain about later restrictions on the root node after adding arms. Thus running $AC_\theta(\theta)$ on a partial triad, does not let us make a statement about a triad derived from it.
- Because of this we define:
- $ACR_{\mathbb{T}}$ names a modification of $AC_{\mathbb{T}}$ that colours the root r that has degree 3 with $L(r) = \{r\}$
- *Rooted core* (RC) names a partial triad θ for which $ACR_\theta(\theta)$ did derive $L(v) = \{v\}$ for every vertex v
- A triad with an arm that is not a RC cannot be a core triad.
- Every partial triad that is not a RC cannot be completed to form a core triad.

5.2.2 Algorithm

$AC \to id$	no statement
$AC \nrightarrow id$	no statement
$ACR \to id$	no statement
$ACR \nrightarrow id$	triad **cannot** be a core

- Consider the arm "100" serves as an example for an arm, on which AC doesn't derive only *id*. Yet, ("100","11","00") is still a core.
- Only if ACR does not derive only *id*, we can drop the arm

5.2.3 Optimizations

- Derive sister triads, e.g. ("01","0","11") from ("10","1","00")
- Optimize arc consistency for endomorphisms (don't check for emtpy lists)

6 TODO Task 4: Commutative Polymorphisms

Write an algorithm that enumerates all core triads that do not have a commutative polymorphism up to a fixed path-length. For every triad $\mathbb{T}$ there is a unique homomorphism

6.1 Notes

- If conjecture is true, then singleton-arc-consistency can be used to check commutative polymorphisms in the same way like path-consistency can be used to check majority polymorphisms
- Singleton-arc-consistency receives the following graph as its input:
  - Calculate the productgraph of $\mathbb{T}$ with itself
  - Merge every pair of vertices (x, y) and (y, x) to one vertex

7 Notes

7.1 Deprecated

7.1.1 Task 1

☒ "3. $\implies$ 1." If $AC_{\mathbb{T}}(\mathbb{T})$ terminates with $L(v) = v$ for all vertices v of $\mathbb{T}$, we know that, if there was a homomorphism $h : \mathbb{T} \to \mathbb{T}$, h would map each vertex v to itself. We see that h is obviously an automorphism, hence $\mathbb{T}$ must be a core.

☒ "1. $\implies$ 2." Our claim is that there has to be a vertex v on p for which $f(v) = v$. To show this we take the orbit of u and the paths in between.

In the simple case we suppose that $f(f(u)) = u$. This implies $f(u_i) = u_{l-i}$ for $i \in \{0, 1, ..., l\}$. Since no double-edges are allowed, we conclude that $l = 2m$, which gives us $f(u_m) = u_m$.

For the general case, we consider the orbit of u to be of size $n \geq 3$. Because of $f(u_0) = u_l$ there is a greatest $m \leq l$ such that $f(u_i) = u_{l-i}$, for every $i \in \{0, 1, ..., m\}$ from which follows that there must be a cyclic path from u_m to $f^n(u_m) = u_m$ of length $n(l - 2m)$. Since $\mathbb{T}$ is a tree, we require that $n(l - 2m) = 0$. The latter equation can only be satisfied for $l = 2m$, and again we get $f(u_m) = u_m$.

7.2 Program

7.2.1 Flags

- **-v** be verbose
- **-l** *NUM* max arm length of triads to generate
- **-p** *NAME* polymorphism to check
- **-t** *STRING* run algorithm only on triad

7.2.2 Ideas

- Store an adjacencylist in every triad

7.3 Todo

7.3.1 TODO Serialize triads

7.3.2 TODO Imlement power$_4$ more efficient

7.3.3 TODO Implement -t flag

-t Triad, Triad being of the form 011010,01010,000

7.3.4 TODO Explain notation in context of triads

e.g. $(p_1 p_2 p_3)$

7.4 Questions

- Do I have to check Siggers-polymorphisms? That f with f(x, y) = f(y, x) suffices is only a claim
- Triads up to a certain number of vertices?
- Triads smaller than $01001111, 0110000, 101000$?
- Triads up to a certain armlength?
- Proof that is understandable for humans and not just based on running a computer program?

7.5 Tasks

- Verify the claims of Jakub Bulin: is it correct that the triad given above does not have a Siggers polymorphism? This can be checked by a computer.
- Write a computer program that generates all triads up to a certain numebr of vertices. Actually, we are only interested in those triads that are *cores*, i.e., not homomorphically equivalent to smaller triads - this greatly reduces the numebr of casse to look at.
- Write a program that verifies Bulin's conjecture on those triads.