Repetitive Symmetries: From Planning Problem Graphs to General Graphs

Florian Sextl
Technische Universität München, Germany

Abstract

We investigate repetitive graph symmetries via the automorphism group and how to utilize thus induced quotient graphs. For this we focus on the notion of descriptive quotients first described in (Abdulaziz, Norrish, and Gretton 2015). We find theoretical and practical results about searching and using these quotients and lay the foundation for future work.

Introduction

Many **NP**-hard computational problems can be translated to equivalent graph problems. This allows to apply well-known graph algorithms and properties to reduce the complexity or improve solutions for the original problems.

One of the more frequently utilized properties is graph symmetry. These symmetries can be exploited to reduce the given graph, and thus the underlying problem, to an equivalent but smaller instance. Following this approach, Abdulaziz et al. leveraged graph symmetries to reduce planning problems in (Abdulaziz, Norrish, and Gretton 2015). The research project this report concludes aimed at lifting and improving the notion of descriptive quotients developed in the aforementioned work. We investigated how descriptive quotients can be found for arbitrary graphs and which properties enable efficient search and optimization procedures based on this concept.

Background

Definitions and Notation

Definition 1 (Groups)**.** *We use the standard definition of a group Φ as a set equipped with an associative operation "$\cdot$" that is closed on Φ, an identity element $\epsilon \in \Phi$ and inverse elements denoted by σ^{-1} for $\sigma \in \Phi$. For a subgroup Γ of Φ we write $\Gamma \leq \Phi$. A group generated by the set of elements $\{\gamma_1, \ldots, \gamma_n\}$ is expressed by $\langle \gamma_1, \ldots, \gamma_n \rangle$. The order or cardinality of the group Φ is denoted by $|\Phi|$.*

Definition 2 (Group Operation and Action)**.** *We primarily work with permutation groups in the following and use the general group operation symbol for permutation composition if not stated otherwise. We denote the application of a permutation ρ to an element v of the underlying structure as $\rho(v)$.*

For two permutations α and β we denote the permutation composition in a right-to-left way as $\beta \cdot \alpha$ to express that the compound acts by first applying α and then β:

$$(\beta \cdot \alpha)(v) = \beta(\alpha(v))$$

We also leave out the operation symbol $\cdot$ for longer chains of composition. Therefore, we denote that two group elements α and β are conjugate with a conjugating element σ as $\beta = \sigma\alpha\sigma^{-1}$.

The orbit of an element v under the action of the group Φ is written as v^Φ. Similarly, the set of all orbits of V under the action of Φ is denoted as V^Φ.

Definition 3 (Graphs)**.** *For the scope of this report, we define a graph G as a tuple (V_G, E_G) of a set of vertices V_G and a set of edges E_G. Our work also focuses only on undirected graphs; thus edges are defined as unordered 2-tuples of vertices. For ease of notation, we define that G^Φ denotes the orbits of Φ's action on the vertices of G. The vertices can be associated with colors to define a colored graph, as well. Yet, the colors are not encoded explicitly but as part of the vertex representation for this report.*

If P is a partition of V_G, we denote the quotient graph induced by P as G/P. We define this quotient graph to comprise the partition's image as the vertex set and the following edge set (with the implicit assumption $v_P \neq w_P$):

$$\{(v_P, w_P) \mid v_P, w_P \in P, \exists v \in v_P, \exists w \in w_P, (v, w) \in E_G\}$$

Definition 4 (Transversals)**.** *We define a transversal to be a function from a set of sets to a representative of a given set. For example could a transversal $\pitchfork$ for G^Φ be defined to map v^Φ to v:*

$$\forall w.w^\Phi = v^\Phi \Longrightarrow \pitchfork(w^\Phi) = v$$

If a transversal of a partition P preserves adjacency in the quotient graph G/P, i.e. if the representatives of two adjacent orbits are also adjacent, then we say that the transversal is consistent with G/P. Based on this, a quotient graph is defined to be consistent if there exists a consistent transversal for it. In addition, we define a quotient graph to be trivially consistent if all vertices of adjacent subsets are also adjacent. In this case, all possible transversals are consistent.

Definition 5 (Descriptive Quotient)**.** *An automorphism of a graph is a permutation of its vertices such that the resulting*

graph is identical to the original one. We denote the automorphism group of the graph G as $\mathcal{A}(G)$.

If the quotient G/G^Γ for $\Gamma \leq \mathcal{A}(G)$ is consistent, it is called descriptive.

Problem Description

For this research project we investigated descriptive quotient graphs and explored their applications. The underlying problem can be defined as:

Problem 1 (DQG[1]). *Given a (possibly colored) graph G, decide whether there exists any non-trivial subgroup Γ of $\mathcal{A}(G)$ that induces a descriptive quotient G/G^Γ.*

To decide this problem, it is necessary to acquire information about the graph's automorphism group and then conduct a search for subgroups that induce (possibly optimal) descriptive quotients. Though, the optimality criteria might differ between applications. For example might smaller quotients be better for some applications while sparser quotients are better for others.

In general, every graph has an optimal descriptive quotient with regard to any metric. Yet, in some cases the optimum might be the trivial quotient induced by the automorphism subgroup consisting of only the identity permutation. The quotient is thus equal to the original graph and, as a result, not of any more use than the original problem. Despite this weakness, it is necessary to include the trivial quotient to make the metrics also well defined for graphs with no other descriptive quotients. This can be the case if the whole automorphism group is trivial or if the induced quotients for all other subgroups contain structures that we call non-descriptive cores, which are described in more detail in a latter section.

SAT Encoding

To investigate descriptive quotients further, we encode whether a quotient graph is consistent as a boolean satisfiability (SAT) problem.

In the following we denote concrete encodings into SAT variables with $[\![\ldots]\!]$. The underlying idea is that a variable can be set to *true* if and only if it either encodes an existing edge in the (quotient) graph or an orbit representative picked by a consistent transversal. The resulting encoding follows the ideas in (Abdulaziz, Norrish, and Gretton 2015) and comprises three different parts:

(i) Encoding of edges:
All edges in the original graph and in the induced quotient graph are encoded to be *true*:

$$\bigwedge_{e \in E_G} [\![e]\!] \wedge \bigwedge_{e' \in E_{G/G^\Gamma}} [\![e']\!] \tag{1}$$

(ii) Encoding of transversals:
For each orbit $o \in G^\Gamma$ a transversal picks exactly one vertex in the orbit as the corresponding representative (denoted as v^o for vertex v picked for orbit o). This is encoded as an "Exactly One" constraint. There exist several encoding approaches for this kind of constraint but the simplest is the pairwise encoding (e.g. Cai et al. 2019):

$$\bigwedge_{o \in G^\Gamma} \left(\bigvee_{v \in o} [\![v^o]\!] \wedge \bigwedge_{v,w \in o, v \neq w} (\neg [\![v^o]\!] \vee \neg [\![w^o]\!]) \right) \tag{2}$$

(iii) Encoding of consistency constraint:
If there exists an edge e' between two orbits $o_1, o_2 \in G^\Gamma$ in the quotient graph, there must also exist an edge e between their representatives $v_1 \in o_1, v_2 \in o_2$:

$$\begin{aligned} \bigwedge_{e' \in E_{G/G^\Gamma}} ([\![e']\!] \wedge [\![v_1^{o_1}]\!] \wedge [\![v_2^{o_2}]\!]) \Longrightarrow [\![e]\!] \equiv \\ \bigwedge_{e' \in E_{G/G^\Gamma}} \neg [\![e']\!] \vee \neg [\![v_1^{o_1}]\!] \vee \neg [\![v_2^{o_2}]\!] \vee [\![e]\!] \end{aligned} \tag{3}$$

Due to the direct encoding of all edges in both the original and the quotient graph for formula (1), every satisfying assignment must map the edge encodings $[\![e]\!]$ and $[\![e']\!]$ to *true*. Thus, all clauses in formula (3) for which e exists in G are trivially true. Similarly, all clauses in formula (3) for which e' does not exist in G/G^Γ are also trivially true. In conclusion, the formula consisting of only formula (2) and formula (3) without the edge encodings (i.e. only $\neg [\![v_1^{o_1}]\!] \vee \neg [\![v_2^{o_2}]\!]$ for existing edges in the quotient but non-existing edges in the original graph) is equisatisfiable to the formula described above.

The resulting formula is then (using again the pairwise encoding for the Exactly One constraint):

$$\begin{aligned} \bigwedge_{o \in G^\Gamma} \left(\bigvee_{v \in o} [\![v^o]\!] \wedge \bigwedge_{v,w \in o, v \neq w} (\neg [\![v^o]\!] \vee \neg [\![w^o]\!]) \right) \wedge \\ \bigwedge_{(o_1,o_2) \in E_{G/G^\Gamma}, v_1 \in o_1, v_2 \in o_2, (v_1,v_2) \notin E_G} \neg [\![v_1^{o_1}]\!] \vee \neg [\![v_2^{o_2}]\!] \end{aligned} \tag{4}$$

We note that the reduced formula requires only a reasonable amount of additional computational effort. The main reason for this is the lookup for edges in the original graph, i.e. whether they exist and thus whether a clause in formula (3) is trivially true. Depending on the exact graph representation, the necessary effort is comparable to iterating the edges for formula (1).

It is worth noting, that all satisfying assignments for formula (4) encode exactly one consistent transversal. Though, if an application does not require a specific consistent transversal, it is possible to reduce the formula even further by changing formula (2) from an "Exactly One" encoding to an "At Least One" encoding. The resulting formula is then:

$$\begin{aligned} \bigwedge_{o \in G^\Gamma} \bigvee_{v \in o} [\![v^o]\!] \quad \wedge \\ \bigwedge_{(o_1,o_2) \in E_{G/G^\Gamma}, v_1 \in o_1, v_2 \in o_2, (v_1,v_2) \notin E_G} \neg [\![v_1^{o_1}]\!] \vee \neg [\![v_2^{o_2}]\!] \end{aligned} \tag{5}$$

[1] Short for "Descriptive Quotient Graph" problem.

Furthermore, this reduction does not change the satisfiability and thus the encoded statement about the existence of consistent transversals:

Lemma 1. *The formulæ* (4) *and* (5) *are equisatisfiable.*

We prove this lemma by contradiction:

Proof. We assume that the formulæ (4) and (5) are not equisatisfiable. Thus, there exist quotient graphs for which only one of them is satisfiable.

We first assume that formula (4) is satisfiable whereas formula (5) is not satisfiable for an arbitrary but fixed quotient graph. We note that formula (5) is a subformula of formula (4) and both are in conjunctive normal form. A satisfying assignment to a formula in conjunctive normal form satisfies also all of its subformulæ. Therefore, the assumed existence of a satisfying assignment for formula (4) implies that the same assignment also satisfies formula (5), which is a contradiction to our assumption.

In a second step we assume that formula (5) is satisfiable whereas formula (4) is not satisfiable for an arbitrary but fixed quotient graph. We note that any satisfying assignment for formula (5) would also satisfy formula (4) if it assigns the value *true* to exactly one variable per orbit. Due to our assumption, there can be no such assignment, but only satisfying assignments for formula (5) such that at least one orbit contains two *true*-encoded vertices. We then also know that the *true*-encoded vertices of two adjacent orbits form a complete bipartite graph due to Lemma 6. As a result, it is possible to extract a consistent transversal from the satisfying assignment by picking only one of the *true*-encoded vertices for each orbit. The so picked vertices are always adjacent if their corresponding orbits are adjacent. As a result, by assigning *true* to only the variables corresponding to this consistent transversal, we obtain a satisfying assignment for formula (4), which contradicts our assumption before.

In conclusion, it is not possible that only one of the formulæ (4) and (5) is satisfiable while the other is unsatisfiable. Thus, the two formulæ are equisatisfiable. □

Search for Descriptive Quotients

To be able to utilize descriptive quotients to their full extend, we need to decide whether they exist for a specific graph and also find such a quotient. In general, a search for a graph's descriptive quotients operates on the space of subgroups of the graph's automorphism group. Therefore, search procedures need to first determine the automorphism group, e.g. by obtaining a generating set for it. In a next step, subgroups must be checked until a descriptive quotient is found for one of them.

As the number of subgroups of the automorphism group can be very high, a linear search through them is not feasible in most cases. Instead, search procedures prune the search space based on heuristics and group theoretic properties. We investigate several such approaches and describe them in greater detail in the following sections.

Conjugacy

We first examine conjugacy of the automorphism group. Thereby, we discover that conjugation of both single automorphisms (Lemma 2) and automorphism subgroups (Theorem 1) preserves consistency of the induced quotient and thus also descriptiveness:

Lemma 2. *If an automorphism induces a descriptive quotient, then also all members of its conjugacy class induce descriptive quotients.*

Proof. We assume that an arbitrary but fixed automorphism α induces a descriptive quotient and lies in the same conjugacy class as a different automorphism β, such that $\beta = \sigma\alpha\sigma^{-1}$ for some automorphism σ.

As α induces a descriptive quotient, there exists a consistent transversal $\pitchfork_\alpha$ on the orbits of α. This allows us to define a transversal $\pitchfork_\beta$ for the orbits of β based on $\pitchfork_\alpha$ by lifting it with σ:

$$\pitchfork_\beta(o_\beta) := \sigma \circ \pitchfork_\alpha(\{\sigma^{-1}(v) \mid v \in o_\beta\})$$

where o_β is an orbit of β and $\circ$ denotes function composition.

We justify this definition by noting that two permutations are conjugate iff they share the same cycle type. In addition, we note that if α moves element x to y, then β moves $\sigma(x)$ to $\sigma(y)$, due to:

$$\begin{aligned}\beta(\sigma(x)) &= (\sigma\alpha\sigma^{-1})(\sigma(x)) = (\sigma\alpha\sigma^{-1}\sigma)(x)\\ &= \sigma(\alpha(x)) = \sigma(y)\end{aligned}$$

Thus, by moving the elements in o_β by σ^{-1}, we get an orbit of α (i.e. $\{\sigma^{-1}(v) \mid v \in o_\beta\}$). Similarly, by moving the representative picked by $\pitchfork_\alpha$ back with σ we get a representative for o_β. As a result, $\pitchfork_\beta$ is in fact a transversal of the orbits of β.

Moreover, $\pitchfork_\beta$ is a consistent transversal. This follows from the fact that $\pitchfork_\alpha$ is consistent. Due to σ^{-1} being an automorphism of the graph, the images of two adjacent orbits $o_{\beta,1}$ and $o_{\beta,2}$ under σ^{-1} are also adjacent. Thus, the two resulting orbits of α, $\{\sigma^{-1}(v) \mid v \in o_{\beta,1}\}$ and $\{\sigma^{-1}(v) \mid v \in o_{\beta,2}\}$, are adjacent as well. Due to this, the representatives picked by $\pitchfork_\alpha$ for these orbits are also adjacent by the definition of consistency. Furthermore, the images of these representatives under the automorphism σ are again adjacent. Hence, $\pitchfork_\beta$ is a consistent transversal as it picks adjacent representatives for adjacent orbits.

In conclusion, β induces a descriptive quotient. □

Theorem 1. *If an automorphism subgroup induces a descriptive quotient, then also all members of its conjugacy class induce descriptive quotients.*

Proof. We assume that Γ_α and Γ_β are conjugate subgroups of $\mathcal{A}(G)$ with $\Gamma_\beta = \sigma\Gamma_\alpha\sigma^{-1}$ for some $\sigma \in \mathcal{A}(G)$. We also assume that Γ_α induces a descriptive quotient for G with the consistent transversal $\pitchfork_\alpha$.

We can then define the same $\pitchfork_\beta$ with regard to $\pitchfork_\alpha$ as in Lemma 2. Due to the well-known fact that the conjugating element of conjugate subgroups defines a bijection between their orbits (see e.g. Holt, Eick, and O'Brien 2005, Proposition 4.9), $\pitchfork_\beta$ is trivially a transversal of the orbits of Γ_β. Moreover, $\pitchfork_\beta$ is also consistent as the reasoning used in the proof for Lemma 2 can be applied here as well.

It thus follows that Γ_β induces a descriptive quotient, too. □

These results allow us to reduce the necessary search space from a possibly super-exponential number of subgroups (cf. Holt 2010) to an exponential one. To be more precise, the number of conjugacy classes of a permutation group with degree $n \geq 4$ is bound by $5^{(n-1)/3}$ (Garonzi and Maróti 2015).

Thus, we define linear search in the set of conjugacy class representatives as our baseline search method. Due to Theorem 1, this procedure is a sound and complete way to determine whether a graph has a descriptive quotient and to obtain such a quotient in the process, as well.

Powerset Search

Our first heuristic search method shifts the search space from the set of conjugacy class representatives to a powerset of generators. More specifically, we check the groups generated by all but the empty subset of the generating set obtained from nauty/Traces. This approach leverages the fact that the combination of two generators is not guaranteed to preserve descriptiveness, but is likely to result in a subgroup from a different conjugacy class.

Albeit the theoretical upper bound for this search is super exponential ($\leq 2^{ld(|\mathcal{A}(G)|)}$, cf. Lemma 4) and thus greater than for the linear search in the class representatives, we have found this approach to be preferable for most planning problems. Graphs from this domain tend to have a single digit number of generators but several magnitudes more conjugacy classes. In addition, the powerset of generators can be obtained with less computational effort than the conjugacy classes. Whereas computing the conjugacy class representatives via the standard "solvable radical"/"trivial fitting" method requires several exponentially complex substeps (Holt 2010), a generating set is obtained from nauty/traces anyway and requires no additional overhead.

Non-Descriptive Cores

Another heuristic search method exploits the so called "non-descriptive cores". These structures are subgraphs of the original graph that contain only vertices from a minimal number of orbits such that there is no consistent transversal for the induced partition. The graphs depicted in fig. 4 and fig. 5 are examples for non-descriptive cores. It is easy to see, that starting from any vertex, there is no path traversing all colors exactly once that would end again at the start vertex. The lack of such paths is equivalent to the lack of consistent partial transversals for the orbits in question.

Due to the reduced formula (5), it becomes clear that non-descriptive cores directly correspond to unsatisfiable cores in our SAT encoding. To be more specific, an unsatisfiable core comprises a number of orbits and their respective "At Least One" clauses as well as the consistency constraints that are impossible to fullfil for these orbits.

We utilize this property to find non-descriptive cores easily and use them as the basis for a constructive search method. For this, we ask the SAT solver[2] to not only decide whether the formula is satisfiable but also return a minimal unsatisfiable subformula (MUS) for negative results.

In a next step, the obtained non-descriptive cores are removed from the quotient graph. To achieve this, we recolor the involved vertices and call again nauty/Traces on the refined graph. For the recoloring it proved to work best if the vertices are all given completely new colors. This guarantees that these vertices will lie in singular orbits for the refined graph. This method is guaranteed to terminate as it either finds a descriptive quotient for the refined graph or will remove all symmetries all together and thus result in the trivial automorphism group.

We also tested other heuristics to remove non-descriptive cores based on composition of generators, but their results were less promising and are thus not included in this report.

Planning Problem Graphs

For our experiments we chose planning problem graphs based on the representation described in (Abdulaziz, Norrish, and Gretton 2015). The motivation behind this was the known benefit that planning problems can gain from descriptive quotients. In addition, the corresponding graphs exhibit a great amount of symmetries that allow to have different search methods find different quotients.

Furthermore, the existing infrastructure from (Abdulaziz, Norrish, and Gretton 2015) can be easily adjusted to use our search methods as a wrapper on top of nauty/Traces. Though, the usefulness of this application was not experimentally verified for all search methods yet and is left for future work.

SNAP Network Graphs

As a second application domain, we chose network graphs from the SNAP dataset (Leskovec and Krevl 2014). Due to our tool's restrictions, the experiments were restricted to undirected graphs only.

In our experiments we found that the network graphs tend to have up to several magnitudes more generators than the planning graphs. Furthermore, we found that most network graphs belong to one of two groups: the first group has mostly trivially consistent quotients, whereas for the second group we did not find any results as nauty/Traces were not able to analyze the graphs in reasonable amounts of time and memory space.

Do to these findings, most of our search methods did either terminate due to running out of resources or find no better quotient than the complete one induced by the whole automorphism group. As a result, the SNAP graphs are not included in the experimental data tables and plots.

Experimental Setup

To determine how feasible and useful our search procedures are, we conducted several experiments. The basis for all of these was a reference implementation in Rust (Sextl 2021). Our implementation comprises all search procedures and

[2]We tested this approach with the kitten tool that comes with kissat.

conducts all necessary steps with the help of other more specialized tools.

In a first step our tool calls nauty/Traces (McKay and Piperno 2014) to find the automorphism group for a given input graph. This step results in a generating set for the group. Based on the group generators, we then conduct the actual search for descriptive quotients. In each search step, we check the current quotient for consistency by encoding it into a SAT formula as described before and call the SAT solver kissat (Biere et al. 2020) on it. The conjugacy class representatives are computed with the GAP system (GAP).

All experimental results were obtained on a cluster of 5 virtual machines from the LRZ Compute Cloud. The machines all had access to 10 Intel(R) Xeon(R) Gold 6148 CPU cores and 45GB of RAM each[3]. The timeout for each run was set to 30 minutes with a memory limit of 6GB for the whole run, of which GAP was allowed at most 4GB.

Experimental Results

We tested 4263 different planing problem graphs in total, of which 938 (about 22%) contained no symmetries. Table 1 contains the detailed numbers of descriptive quotients found per method per problem domain. Whereas the top row summarizes the results for all graphs, the middle rows break these down for the underlying problem domains. The bottom rows show then a small selection of results for individual planning problems.

We compared our three search methods against the quotient induced by the whole automorphism group (called "Full Quotient" in the table) as a baseline. It becomes apparent that for most of the planning problem graphs (about 82%), the quotient induced by the whole automorphism group is already descriptive. For these no search is needed, but the quotient obtained from nauty/Traces already suffices.

Yet, our search methods make it possible to find descriptive quotients for up to 88% of the graphs. Both searching in the powerset of generators and removing non-descriptive cores by recoloring perform equally well for this task for all domains. In direct comparison to the search heuristics, the complete search through the conjugacy class representatives did not perform well. It successfully terminated for only 23% of the graphs. In all other cases it either reached the memory limit or a timeout occurred. Yet, the conjugacy class search method was also able to outperform the full quotient method for a few domains (e.g. *barman*).

Other than the success rate, we also investigated the runtime and size of found descriptive quotients. The results for these metrics is recorded in figs. 2 and 3 respectively. In these figures, we compare the powerset heuristic against the conjugacy class search as well as the two heuristics against each other. For the runtime on the one hand, we specifically marked the timeout of 30 minutes and set this value also for earlier terminations due to the memory limit. As the full quotient heuristic is checked as part of the other search methods, it takes at least as much time as these and is thus not included in our results. On the other hand, the quotient sizes are measured as the ratio to the original graph's size. As a result, all not terminating experiments are given the ratio of 1 as the only found descriptive quotient can be the original graph itself. Similar to the runtime plot, we omitted the full quotient method again, as the so found quotients have the lowest size ratio by default, because they utilize all symmetries.

We conclude based on the plotted data that the conjugacy class search always takes longer and results on average in larger quotients than our heuristics. Of these heuristics, the core recoloring takes on average more time than the powerset search, yet finds significantly smaller quotients. Thus, both heuristics are advantageous over the conjugacy class search and also reasonable fallback methods for graphs with non-descriptive full quotients.

[3]This corresponds to the lrz.xlarge flavor of virtual machines as described here: https://doku.lrz.de/display/PUBLIC/Compute+Cloud

Other Theoretical Aspects

Apart from the search procedures, we also strive to provide an overview of the computational complexity of DQG. We first note a more general result:

Lemma 3 (**Abdulaziz, Norrish, and Gretton 2015**). *It is **NP**-complete to decide whether a consistent transversal exists for a given vertex partition.*

Due to the additional requirement of the partition consisting of automorphism orbits, we can not directly transfer this result to DQG, though. We thus first start with the following theorem:

Theorem 2. *The descriptive quotient graph problem DQG lies in **NP**.*

Proof. The certificate for the graph G comprises both a minimal generating set of a subgroup $\Gamma \leq \mathcal{A}(G)$ and a fitting consistent transversal $\pitchfork$ for the quotient G/G^Γ.

To check whether the generated group is actually a subgroup of $\mathcal{A}(G)$, it is sufficient to check whether all generators are automorphisms. To in turn check whether a permutation ρ is an automorphism, it suffices to check the graph isomorphism problem GI for the input (G, G) with the certificate ρ. This is possible in deterministic polynomial-time as GI is known to lie in **NP**.

It is also possible to compute the quotient G/G^Γ induced by Γ in polynomial time. For this, at first the subgroup action's orbits are computed inductively. The algorithm for this starts with a discrete partition and then stepwise unifies orbits if a generator permutes vertices between them. By running this procedure iteratively for all permutations in the generating set, we get the induced partition and can add the corresponding edges to obtain the whole quotient in polynomial time.

In addition, it is also possible to check whether this quotient is descriptive by using the transversal as a certificate and running the polynomial-time decision procedure which we get from Lemma 3.

To conclude, the certificate comprising a generating set and a consistent transversal has both polynomial size (cf. Lemma 5) and can be checked in deterministic polynomial

Domain	Number of Graphs	Full Quotient	Conjugacy Class Search	Powerset Search	Core Recoloring
All Graphs	3325	2743	765	2939	2951
IPC5	208	143	47	177	175
IPC8	434	201	90	227	225
gripper	22	21	5	21	21
logistics_Track	397	312	317	318	321
seq	653	206	107	211	213
tseq	90	34	46	50	54
unsolve	939	348	111	453	459
Test2	40	20	6	20	21
newopen	1440	1440	14	1440	1440
zeno	40	18	22	22	22
hiking	40	40	0	40	40
pipesworld	46	14	14	44	42
storage	30	0	1	3	3
barman	62	18	43	46	49

Figure 1: Number of descriptive quotients found per domain and search method

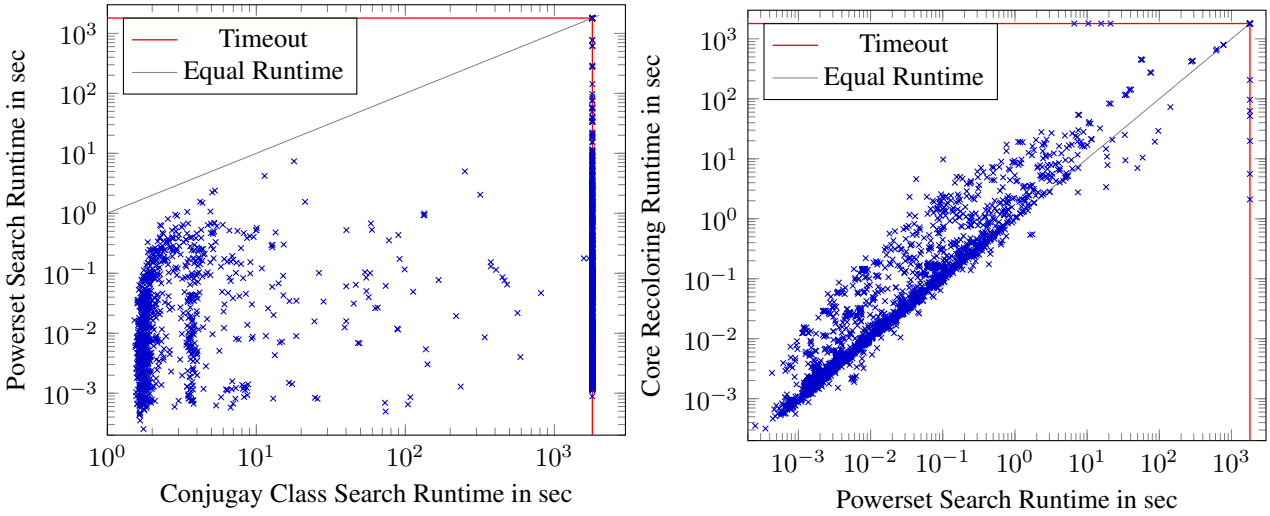

Figure 2: Comparison of search runtimes

time. It can also be noted that a graph has a non-trivial subgroup inducing a descriptive quotient iff there exists such a certificate by definition.

Thus, we have shown that DQG $\in$ **NP**. □

Based on the decision problem 1, it is possible to define the following optimization problem:

Problem 2 (DQG-Opt)**.** *Given a graph G and a metric $\mathcal{M}$, find $\Gamma \leq \mathcal{A}(G)$ that induces an optimal descriptive quotient G/G^Γ with respect to $\mathcal{M}$. This means that $\mathcal{M}(G/G^\Gamma) = max\{\mathcal{M}(G/G^{\Gamma'}) \mid \Gamma' \leq \mathcal{A}(G),\ G/G^{\Gamma'}$ descriptive$\}$.*

This problem is parameterized by a metric to categorize different descriptive quotients and which we define to be maximal for an optimal quotient. Following this definition, is is also possible to define approximation algorithms based on the aforementioned search procedures.

As mentioned before, it is always possible to find an optimal descriptive quotient by including the trivial one into the search space. Yet, this restricts the used metrics to those that do not become maximal for the trivial group. Otherwise, the metric would be of no use as the optimal quotient graph would be the same as the original one.

Summary, Future and Related Work

In this report we described the results of our research on descriptive quotients and their use cases. We confirmed that even simple heuristics such as searching in the powerset of generators can be used to improve applications of descriptive quotients. In addition, we determined several properties of descriptiveness which allow for more advanced search approaches. We expect, for example, to find more heuristics to remove non-descriptive cores without destroying all symmetries in the process.

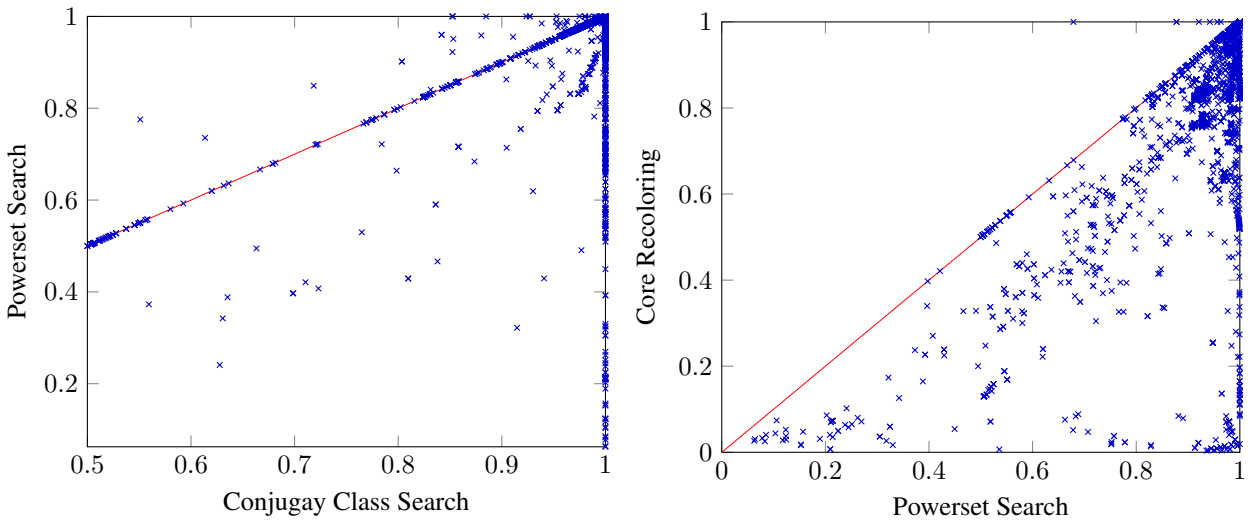

Figure 3: Comparison of size ratio between found quotients and original graphs

On another note, it might be possible to combine the construction of $\mathcal{A}(G)$ by nauty/Traces with a search procedure for descriptive quotients as the Schreier-Sims algorithm, which forms the basis of the graph symmetry analysis, utilizes similar results from group theory as our conjugacy class search method. A related idea can also be found in the GRAPE package (Soicher 2021) for the GAP system (GAP). This package combines reasoning about graphs with group computations, yet does not support descriptive quotient graphs as of the time of this writing.

Whereas we focused on graph automorphisms, Horčík et al. employed the more general endomorphisms to reduce labeled transition systems for planning problems in their recent work (Horčík and Fišer 2021). Their basic idea was to remove redundant actions with higher costs. Similar to our SAT encoding approach, they encoded the endomorphism properties into a constraint satisfaction problem. Yet, as the endomorphism exploits planning actions and their costs, this approach is not trivially liftable to general graphs and thus resembles more the work of Abdulaziz et al. in (Abdulaziz, Norrish, and Gretton 2015) than our research scope.

To summarize, there are still several open questions and improved approaches left to investigate based on our findings. Especially the correlation of graph symmetry and group properties seems to allow for promising results in the future.

References

Abdulaziz, M.; Norrish, M.; and Gretton, C. 2015. Exploiting symmetries by planning for a descriptive quotient. In *Proc. of the 24th International Joint Conference on Artificial Intelligence, IJCAI*, 25–31.

Biere, A.; Fazekas, K.; Fleury, M.; and Heisinger, M. 2020. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling Entering the SAT Competition 2020. In Balyo, T.; Froleyks, N.; Heule, M.; Iser, M.; Järvisalo, M.; and Suda, M., eds., *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, 51–53. University of Helsinki.

Cai, J.; Su, Y.; Yang, X.; Min, J.; and Lai, Y. 2019. A new SAT Encoding Scheme for Exactly-one Constraints. *Journal of Physics: Conference Series* 1288: 012035. doi:10.1088/1742-6596/1288/1/012035.

GAP. 2020. GAP – Groups, Algorithms, and Programming, Version 4.11.0. https://www.gap-system.org.

Garonzi, M.; and Maróti, A. 2015. On the number of conjugacy classes of a permutation group. *Journal of Combinatorial Theory, Series A* 133: 251–260. doi:10.1016/j.jcta.2015.02.007.

Geyer, L. 2012. Why is the minimum size of a generating set for a finite group at most $\log_2 n$? Mathematics Stack Exchange. URL https://math.stackexchange.com/q/226938.

Holt, D. F. 2010. Enumerating subgroups of the symmetric group. *Computational Group Theory and the Theory of Groups, II* 511: 33–37.

Holt, D. F.; Eick, B.; and O'Brien, E. A. 2005. *Handbook of Computational Group Theory*. Discrete mathematics and its applications. Chapman & Hall/CRC. doi:10.1201/9781420035216.

Horčík, R.; and Fišer, D. 2021. Endomorphisms of Classical Planning Tasks. *Proceedings of the AAAI Conference on Artificial Intelligence* 35(13): 11835–11843. ISSN 2374-3468. URL https://ojs.aaai.org/index.php/AAAI/article/view/17406.

Leskovec, J.; and Krevl, A. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

McKay, B. D.; and Piperno, A. 2014. Practical graph iso-

morphism, {II}. *Journal of Symbolic Computation* 60: 94 – 112. ISSN 0747-7171. doi:10.1016/j.jsc.2013.09.003.

Sextl, F. 2021. Descriptive Quotient Graphs Tool. URL https://github.com/firefighterduck/dqg.

Soicher, L. H. 2021. GRAPE, GRaph Algorithms using PErmutation groups, Version 4.8.5. https://gap-packages.github.io/grape. Refereed GAP package.

Appendices

Proofs

We only consider finite graphs in the following as these are the only relevant with regard to computational complexity.

Lemma 4. *Every minimally generated group* Γ *has a minimal generating set* $gen(\Gamma) := \{\gamma_1, \dots, \gamma_m\}$ *of cardinality* $|gen(\Gamma)| = m \leq ld(|\Gamma|)$.

The following proof for Lemma 4 is based on a proof idea from (Geyer 2012).

Proof. We do an induction proof on the group generated by generators γ_1 to γ_i (i.e. $\langle \gamma_1, \dots, \gamma_i \rangle$). We also note that because gen(Γ) is minimal all generators are not the identity permutation ϵ.

- Base case $i = 1$:
  We need to show $|\langle \gamma_1 \rangle| \geq 2^1 = 2$. Due to $\gamma_1 \neq \epsilon$ the generated group $\langle \gamma_1 \rangle$ needs to contain at least ϵ and γ_1 and has thus order of at least 2.
- Induction step:
  We assume that for a fixed but arbitrary i, $|\langle \gamma_1, \dots, \gamma_i \rangle| \geq 2^i$ holds. We then show that $|\langle \gamma_1, \dots, \gamma_{i+1} \rangle| \geq 2^{i+1}$ also holds. We first note that due to the minimality of the generating set, no generator can be obtained by composing any string of other generators and their inverses. As a result, it holds that $\gamma_{i+1} \notin \langle \gamma_1, \dots, \gamma_i \rangle$. It follows that the left cosets $\epsilon\langle \gamma_1, \dots, \gamma_i \rangle = \langle \gamma_1, \dots, \gamma_i \rangle$ and $\gamma_{i+1}\langle \gamma_1, \dots, \gamma_i \rangle$ are not identical as only the second one contains γ_{i+1}. Because both sets are cosets of $\langle \gamma_1, \dots, \gamma_i \rangle$, they thus must be disjunct and also of the same size as $\langle \gamma_1, \dots, \gamma_i \rangle$. Therefore, we can follow that $\langle \gamma_1, \dots, \gamma_{i+1} \rangle$ contains both cosets and has thus order of at least $2 \cdot |\langle \gamma_1, \dots, \gamma_i \rangle|$ which by induction hypothesis results in $|\langle \gamma_1, \dots, \gamma_{i+1} \rangle| \geq 2 \cdot 2^i = 2^{i+1}$.

  As a result, $|\langle \gamma_1, \dots, \gamma_m \rangle| = |\Gamma| \geq 2^m$ and $m \leq ld(|\Gamma|)$. □

With this result, we can also proof an upper bound for the size of minimal generating sets of subgroups of the automorphism group:

Lemma 5. *Every subgroup* Γ *of the automorphism group* $\mathcal{A}(G)$ *has a generating set with order polynomial in the size of the graph* $n := |V_G|$.

Proof. Every subgroup Γ of the automorphism group $\mathcal{A}(G)$ is finite and is thus also finitely generated. As a consequence, Γ is minimally generated by a finite minimal generating set.

We can thus pick an arbitrary but fixed minimal generating set for Γ comprising m generators. Due to Lemma 4, we know that $m \leq ld(|\Gamma|)$.

In the worst case, Γ can be as large as the whole symmetric group S_n, which has order $n!$. We can thus find the following bound for m:

$$m \leq ld(|\Gamma|) \leq ld(n!) = ld\left(\prod_{i=1}^{n} i\right) =$$
$$= \sum_{i=1}^{n} ld(i) = \frac{n(n-1)}{2} \leq n^2$$

Thus, Γ has a minimal generating set of order polynomial in the graph size n. □

Lemma 6. *For every satisfying assignment to formula* (5)*, for every two adjacent orbits* o_1 *and* o_2*, all true-encoded vertices in* o_1 *are adjacent to all true-encoded vertices in* o_2 *and vice versa.*

Proof. Without loss of generality, we assume that the vertex encodings $[\![v_1^{o_1}]\!]$ and $[\![v_2^{o_1}]\!]$ are assigned to *true* and only v_2 is adjacent to v_3 with $[\![v_3^{o_2}]\!]$ assigned to *true*. Then the consistency constraints contains a clause $\neg[\![v_1^{o_1}]\!] \vee [\![\neg v_3^{o_2}]\!]$ which prevents that both $[\![v_1^{o_1}]\!]$ and $[\![v_3^{o_2}]\!]$ can be assigned *true*. This is a contradiction to our assumption and as a result, all *true*-encoded vertices in o_1 need to be adjacent to all *true*-encoded vertices in o_2. □

Example Graphs

The following graphs depict non-descriptive subgraphs obtained from planning problems. Vertices of the same color form an orbit in the corresponding quotient graph. For all graphs exists no consistent transversal that would pick one vertex of each color (i.e. each orbit) and keep all inter-color edges intact.

All of the following non-descriptive subgraphs were taken from the planning problems used for our experiments.

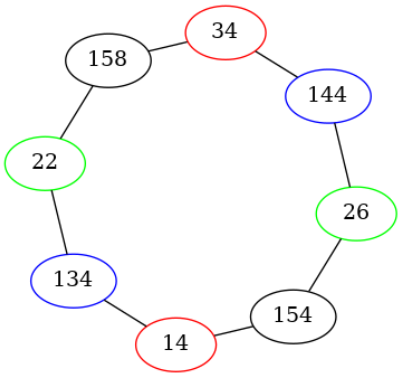

Figure 4: Nondescriptive subgraph for planning problem IPC5_storage_p03.

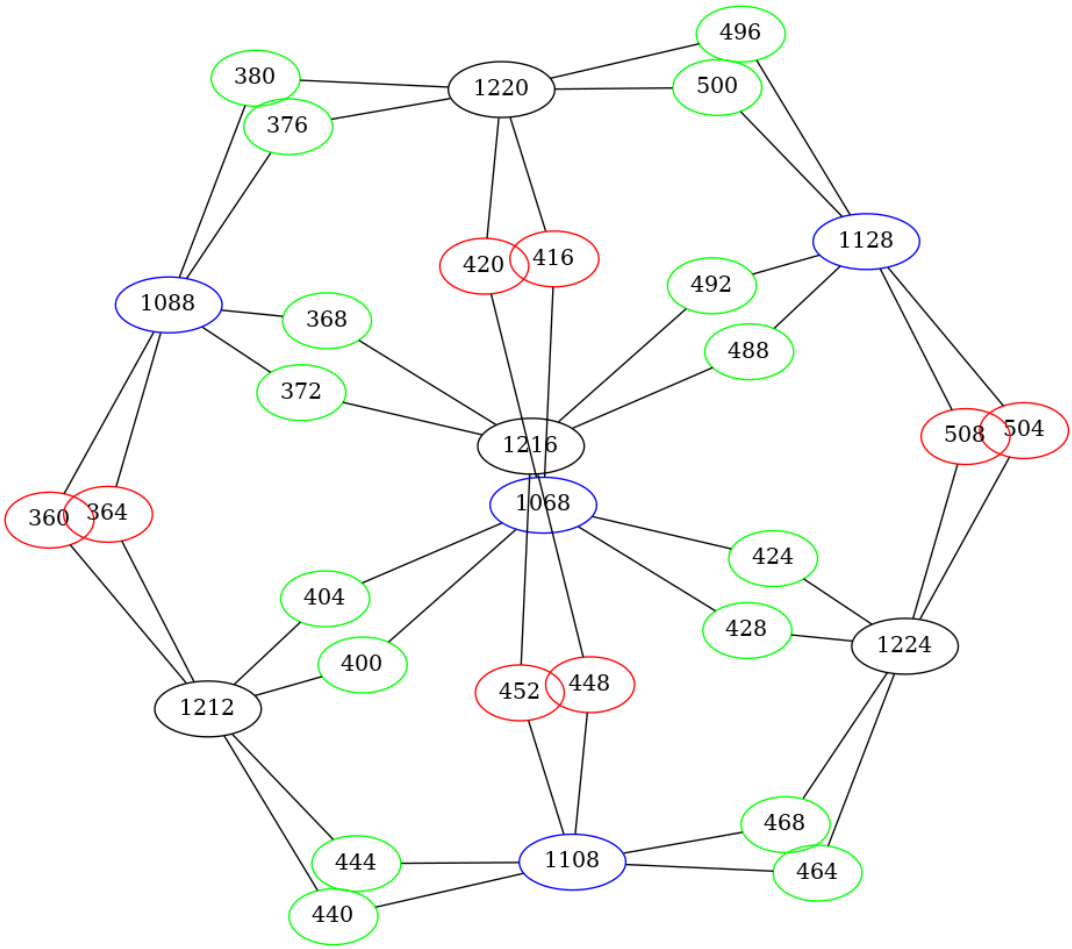

Figure 5: Nondescriptive subgraph for planning problem IPC8_opt_Barman_p435.1.