LIE ALGEBRA CONJUGACY

JOSHUA A. GROCHOW
UNIVERSITY OF CHICAGO
JOSHUAG@CS.UCHICAGO.EDU

ABSTRACT. We study the problem of matrix Lie algebra conjugacy. Lie algebras arise centrally in areas as diverse as differential equations, particle physics, group theory, and the Mulmuley–Sohoni Geometric Complexity Theory program. A matrix Lie algebra is a set $\mathcal{L}$ of matrices such that $A, B \in \mathcal{L} \implies AB - BA \in \mathcal{L}$. Two matrix Lie algebras are conjugate if there is an invertible matrix M such that $\mathcal{L}_1 = M\mathcal{L}_2 M^{-1}$.

We show that certain cases of Lie algebra conjugacy are equivalent to graph isomorphism. On the other hand, we give polynomial-time algorithms for other cases of Lie algebra conjugacy, which allow us to mostly derandomize a recent result of Kayal on affine equivalence of polynomials. Affine equivalence is related to many complexity problems such as factoring integers, graph isomorphism, matrix multiplication, and permanent versus determinant.

Specifically, we show:
- Abelian Lie algebra conjugacy is as hard as graph isomorphism. A Lie algebra is abelian if all of its matrices commute pairwise.
- Abelian diagonalizable Lie algebra conjugacy of $n \times n$ matrices can be solved in $poly(n)$ time when the Lie algebras have dimension $O(1)$. The dimension of a Lie algebra is the maximum number of linearly independent matrices it contains. A Lie algebra $\mathcal{L}$ is diagonalizable if there is a single matrix M such that for every A in $\mathcal{L}$, MAM^{-1} is diagonal.
- Semisimple Lie algebra conjugacy is equivalent to graph isomorphism. A Lie algebra is semisimple if it is a direct sum of simple Lie algebras.
- Semisimple Lie algebra conjugacy of $n \times n$ matrices can be solved in polynomial time when the Lie algebras consist of only $O(\log n)$ simple direct summands.
- Conjugacy of completely reducible Lie algebras—that is, a direct sum of an abelian diagonalizable and a semisimple Lie algebra—can be solved in polynomial time when the abelian part has dimension $O(1)$ and the semisimple part has $O(\log n)$ simple direct summands.

1. INTRODUCTION

A *matrix Lie algebra* is defined as a set of $n \times n$ matrices closed under the following operations: multiplication by scalars, the usual matrix addition, and a multiplication-like operation denoted $[A, B] := AB - BA$. Lie algebras are an important tool in areas as diverse as differential equations [Olv93, Ste07], particle physics [Geo82], group theory [FH91, Che46, OV90], and the Mulmuley–Sohoni Geometric Complexity Theory program [MS01].

In complexity theory, Kayal [Kay11a] has recently used Lie algebras in the so-called affine equivalence problem, which arises in many areas of complexity: factoring integers, permanent versus determinant, matrix multiplication, lower bounds for depth-three circuits, and several more (see [Kay11a, §1.1]). Kayal essentially used Lie algebra conjugacy to give a randomized polynomial-time

---

algorithm to decide when a function can be gotten from the determinant by an invertible linear change of variables. This is the affine equivalence problem for the determinant.

The following are examples of Lie algebras, which should help give their flavor, and introduces some of those Lie algebras on which we prove results, namely abelian, diagonalizable, and (semi-)simple:

(1) The collection of all $n \times n$ matrices.

(2) The collection of all diagonal $n \times n$ matrices is a Lie algebra of dimension n. Any two diagonal matrices D_1, D_2 commute. Since $D_1 D_2 - D_2 D_1 = 0$ this is a Lie algebra. Any Lie algebra in which all matrices commute is called *abelian*.

(3) In fact, *any* collection of diagonal matrices that is closed under taking linear combinations is a Lie algebra, for the same reason as above. Furthermore, if $\mathcal{D}$ is such a Lie algebra, then $A\mathcal{D}A^{-1}$ is as well, since conjugating by A preserves the fact that all the matrices in $\mathcal{D}$ commute. Any Lie algebra conjugate to a set of diagonal matrices is called *diagonalizable*.

(4) The collection of all $n \times n$ matrices with trace zero. Since $\operatorname{tr}(AB - BA) = 0$ for any A, B, this is also a Lie algebra. This is an example of a *simple* Lie algebra.

(5) The collection of all $2n \times 2n$ matrices of the form $\begin{pmatrix} C & 0 \\ 0 & D \end{pmatrix}$ where C, D are $n \times n$ matrices and $\operatorname{tr} C + \operatorname{tr} D = 0$.

Two Lie algebra $\mathcal{L}_1, \mathcal{L}_2$ are *conjugate* if there is an invertible matrix A such that $\mathcal{L}_1 = A\mathcal{L}_2 A^{-1}$, that is, for every matrix M in $\mathcal{L}_2$, AMA^{-1} is in $\mathcal{L}_1$, and $\mathcal{L}_1$ and $\mathcal{L}_2$ have the same dimension. Although it was not phrased this way, Kayal gave a randomized reduction from the affine equivalence problem for the determinant to the Lie algebra conjugacy problem for Lie algebras isomorphic to example (5). However, where he uses properties very specific to the Lie algebras associated to permanent and determinant that can be computed using randomization, we are able to instead use a deterministic approach to the more general problem of Lie algebra conjugacy.

1.1. Results.
We show that certain cases of Lie algebra conjugacy are solvable in polynomial time. We also show that extending these cases is difficult, as such an extension is equivalent to graph isomorphism in one case and at least as hard as graph isomorphism in the other case. One of these cases is strong enough to mostly derandomize Kayal's result [Kay11a] on testing affine equivalence to the determinant (see §5 for details).

We now give the formal definition of Lie algebra conjugacy. Since Lie algebras are closed under taking linear combinations, we can give them as input to algorithms by providing a linear basis.

> **Problem:** Lie Algebra Conjugacy (LAC)
> **Input:** Two Lie algebras $\mathcal{L}_1$ and $\mathcal{L}_2$ of $n \times n$ matrices, given by basis elements.
> **Output:** An invertible $n \times n$ matrix A such that $A\mathcal{L}_1 A^{-1} = \mathcal{L}_2$, if such A exists, otherwise "the Lie algebras are not conjugate."

Recall the definitions of abelian and diagonalizable from examples (2) and (3) above, respectively.

Theorem 2.2. *Abelian diagonalizable Lie algebra conjugacy of $n \times n$ matrices can be solved in $\operatorname{poly}(n)$ time when the Lie algebras have dimension $O(1)$.*

Abelian Lie algebras are one of two fundamental building blocks of all Lie algebras. The other fundamental building blocks are the semisimple Lie algebras. A Lie algebra is semisimple if it is a direct sum of simple Lie algebras. Example (4) above is a simple Lie algebra; see Appendix A for the full definition, and the discussion leading up to Remark A.8 for what we mean by "building blocks." For this other building block, we show a similar result:

Theorem 3.6. *Semisimple Lie algebra conjugacy of $n \times n$ matrices can be solved in $\operatorname{poly}(n)$ time when the Lie algebras have only $O(\log n)$ simple direct summands.*

Despite the $O(\log n)$ restriction, Theorem 3.6 is already strong enough to mostly derandomize Kayal's result (Corollary 5.1 below). Also, note that even a single simple Lie algebra can have unbounded dimension, as in example (4), let alone a semisimple one with $O(\log n)$ simple summands. Theorem 3.7 gives another class on which Lie algebra conjugacy is solvable in polynomial time.

For both results above, we show that removing the quantitative restrictions is likely to be difficult:

Theorem 2.1. *Graph isomorphism polynomial-time reduces to abelian diagonalizable Lie algebra conjugacy, when the Lie algebras may have unbounded dimension.*

Theorem 3.1. *Graph isomorphism is equivalent to semisimple Lie algebra conjugacy, when the Lie algebras may contain an unbounded number of simple direct summands.*

In fact, we show that abelian diagonaliable Lie algebra conjugacy is equivalent to the code equivalence problem. The code equivalence problem is to test whether two subspaces of a vector space can be made equal by permuting their coordinates. For codes over $\mathbb{F}_2$, code equivalence was known to be as hard as graph isomorphism [PR97]; we extend their proof to show that code equivalence over *any field* is as hard as graph isomorphism.

Finally, we combine the abelian and semisimple cases together, to show results on Lie algebra conjugacy when the Lie algebras are the direct sum of an abelian Lie algebra and a semisimple one:

Theorem 4.2. *Conjugacy of Lie algebras of $n \times n$ matrices can be determined in poly(n) time when the Lie algebras are a direct sum of an $O(1)$-dimensional abelian diagonalizable Lie algebra and a semisimple Lie algebra with $O(\log n)$ simple direct summands.*

Since abelian is a special case of abelian-plus-semisimple, this more general case is obviously as hard as code equivalence when we drop the quantitative restrictions of the above theorem.

1.2. **Outline.** In §2 we prove Theorems 2.2 and 2.1 on abelian LAC; this section can be understood without any background on Lie algebras. Our other results require more knowledge of Lie algebras; we collect the necessary background in Appendix A. In §3 we prove Theorems 3.1, 3.6, and 3.7 on semisimple LAC. In §4 we prove our results on direct sums of abelian and semisimple Lie algebras, including Theorem 4.2. In §5 we show how to use the above machinery to mostly derandomize Kayal's result on testing affine equivalence to the determinant. In the final section, we discuss how close the abelian-plus-semisimple case is to the general case, directions toward the general case, and other future work, including potential ways to solve important special cases of the affine equivalence problem efficiently without having to efficiently solve GI.

2. ABELIAN DIAGONALIZABLE LIE ALGEBRA CONJUGACY AND CODE EQUIVALENCE

In this section we show that conjugacy of abelian diagonalizable matrix Lie algebras is Karp-equivalent to the code equivalence problem, and hence is at least as hard as GI. The reduction to code equivalence allows us to solve abelian diagonalizable LAC for constant-dimensional Lie algebras in polynomial time.

A d-dimensional *code* of length n over a field $\mathbb{F}$ is a d-dimensional subspace of $\mathbb{F}^n$. Codes are represented algorithmically by giving bases for them as subspaces. The symmetric group S_n acts on $\mathbb{F}^n$ by permutation of coordinates: for $\pi \in S_n$ and $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}^n$, $\pi \cdot \vec{\alpha} = (\alpha_{\pi(1)}, \ldots, \alpha_{\pi(n)})$. S_n then acts on a subspace $V \subseteq \mathbb{F}^n$ by $\pi \cdot V = \{\pi \cdot v : v \in V\}$. The *code equivalence problem* is: given two codes C_1, C_2, determine whether there is a permutation $\pi \in S_n$ such that $\pi \cdot C_1 = C_2$.

Theorem 2.1. *Abelian dagonalizable Lie algebra conjugacy for d-dimensional subspaces of $n \times n$ matrices is equivalent to code equivalence for d-dimensional subspaces of $\mathbb{F}^n$, for any field $\mathbb{F}$.*

Before proving this theorem we give some of its consequences.

Corollary 2.2. *Conjugacy of abelian diagonalizable $O(1)$-dimensional Lie algebras of $n \times n$ matrices over any field can be solved in poly(n) time.*

Proof. Babai (see [BCGQ11, Theorem 7.1]) showed that, over any field $\mathbb{F}$, equivalence of d-dimensional linear codes of length n reduces to $\binom{n}{d}$ instances of $d \times (n - d)$ edge-colored bipartite graph isomorphism. Each such instance can be solved in $poly(n) \cdot \min\{d!, (n - d)!\}$ time, so when $d = O(1)$ code equivalence can be solved in polynomial time. By Theorem 2.1, d-dimensional diagonalizable Lie algebra conjugacy can be solved in polynomial time when $d = O(1)$.   □

Corollary 2.3. *Abelian diagonalizable Lie algebra conjugacy over any field is* GI-*hard.*

Proof. Petrank and Roth [PR97] showed that GI Karp-reduces to code equivalence over $\mathbb{F}_2$. Over an arbitrary field we use the same reduction, but an extension of their proof is required, which we give in Lemma 2.4 below. Theorem 2.1 then shows that abelian diagonalizable LAC is GI-hard.   □

Lemma 2.4. *Graph isomorphism Karp-reduces to code equivalence over any field* $\mathbb{F}$.

Proof. Given a graph G, we construct the generator matrix for a code over $\mathbb{F}$ such that two graphs are isomorphic if and only if the codes are equivalent. Let $M(G) = [I_m | I_m | I_m | D]$ where $m = |E(G)|$, I_m is the $m \times m$ identity matrix, and D is the incidence matrix of G:

$$D_{e,v} = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{otherwise} \end{cases}$$

The *Hamming weight* of a vector over $\mathbb{F}$ is the number of non-zero entries. The following claim is essentially the crux of Petrank and Roth's argument, but generalized so as to apply over any field.

   Claim: up to permutation and scaling of the rows, $M(G)$ is the unique generator matrix of its code which satisfies the following properties:

   (1) it is a $|E| \times (3|E| + |V|)$ generator matrix;
   (2) each row has Hamming weight ≤ 5;
   (3) any nondegenerate linear combination of two or more rows has Hamming weight ≥ 6

A linear combination of k rows is nondegenerate if all k of its coefficients are nonzero.

   Proof of claim: First, $M(G)$ satisfies (1)–(3). The only part to check is (3): in the first $3m$ columns, any nondegenerate linear combination of $k \geq 2$ rows will have $3k \geq 6$ nonzero entries. Next, let C denote the code generated by the rows of $M(G)$. By (2) and (3) the rows of $M(G)$ are the *unique* vectors in C (up to scaling) of Hamming weight ≤ 5. Hence if M' is any other generator matrix of C satisfying (1)–(3), its rows must be scaled versions of the rows of $M(G)$ in some order. This proves the claim.

   Now, suppose that $M(G_1)$ and $M(G_2)$ generate equivalent codes. Then there is a nonsingular matrix S and a permutation matrix P such that $M(G_1) = SM(G_2)P$. By the claim, $S = \Delta S'$ where Δ is diagonal and S' is a permutation matrix. However, since the first $3|E|$ columns of $M(G_1)$ and $M(G_2)$ only contain $0, 1$-entries, $\Delta = I$. The rest of the proof of the reduction, including the other direction, proceeds exactly as in Petrank and Roth [PR97].   □

Proof of Theorem 2.1. Let $(A_1, \ldots, A_d), (B_1, \ldots, B_d)$ be an instance of abelian diagonalizable LAC. Standard techniques in linear algebra can be used to simultaneously diagonalize the A_i in polynomial time, so we may now assume that the A_i are in fact diagonal, rather than merely diagonalizable. Similarly for the B_i. Let $\mathcal{A}$, resp. $\mathcal{B}$, denote the Lie algebras spanned by the A_i, resp. B_i.

   Claim: If $\mathcal{A}$ and $\mathcal{B}$ are diagonal, then they are conjugate if and only if they are conjugate by a permutation matrix.

   By "flattening out" the entries of the diagonal matrices into "row" vectors the claim shows that diagonalizable d-dimensional LAC of $n \times n$ matrices is Karp-equivalent to d-dimensional code equivalence of codes of length n. Thus the claim will complete the proof of the theorem.

   Proof of claim: Suppose $g\mathcal{A}g^{-1} = \mathcal{B}$. Since $\mathcal{B}$ is diagonal, g must preserve the eigenspaces of every matrix in $\mathcal{A}$. The formalization of this notion will allow us to prove our claim. Let $\lambda_i \colon \mathcal{A} \to \mathbb{F}$ be the linear function $\lambda_i(A) = A_{ii}$. We can think of λ_i as a "simultaneous eigenvalue for the space $\mathcal{A}$ of matrices," generalizing the notion of an eigenvalue of a single matrix. Such functions are called

weights in the theory of Lie algebras, and they will play a significant role here and in the case of semisimple Lie algebras as well. Analogous to an eigenspace corresponding to an eigenvalue, there are *weight spaces* corresponding to weights. Namely, if $\lambda \colon \mathcal{A} \to \mathbb{F}$ is a weight, the corresponding weight space is

$$V_\lambda(\mathcal{A}) := \{v \in \mathbb{F}^n : Av = \lambda(A)v \text{ for all } A \in \mathcal{A}\}$$

It is these weight spaces that g must preserve in order for $g\mathcal{A}g^{-1}$ to be diagonal. For example, if every weight space is 1-dimensional—or equivalently, if for every pair of indices $1 \le i < j \le n$ there is some matrix $A \in \mathcal{A}$ with $A_{ii} \ne A_{jj}$—then g must be a permutation matrix.

More generally, g may send $v \in V_{\lambda_1}$ into V_{λ_2} if and only if $gV_{\lambda_1} = V_{\lambda_2}$. Within each weight space, g may act in an arbitrary invertible manner. In other words, g is composed of invertible blocks of dimension $\dim V_{\lambda_i}$, the pattern in which these blocks appear is a permutation, and that permutation may send $i \mapsto j$ if and only if $\dim V_{\lambda_i} = \dim V_{\lambda_j}$. However, if g' has the same permutation pattern as g but all the blocks in g' are the identity, then $g\mathcal{A}g^{-1} = g'\mathcal{A}g'^{-1}$. Hence, without loss of generality, we may take g to be a permutation matrix, proving the claim. □

3. Semisimple Lie algebra conjugacy and graph isomorphism

Theorem 3.1. *Semisimple Lie algebra conjugacy is equivalent to graph isomorphism.*

Proof. We break the proof into four lemmas. By Lemma 3.2, semisimple Lie algebra conjugacy is equivalent to deciding whether two representations of a semisimple Lie algebra are equivalent up to outer automorphism. By Lemma 3.3, the latter problem reduces to a special case of twisted code equivalence with multiplicities, which we refer to as Problem A. Finally, Lemma 3.4 reduces Problem A to graph isomorphism, and Lemma 3.5 reduces graph isomorphism to semisimple Lie algebra conjugacy. □

Lemma 3.2 (de Graaf[1])**.** *Semisimple Lie algebra conjugacy is equivalent to—nearly just a restatement of—the following problem (see Appendix A.4 for definitions):*

> **Problem:** *Outer equivalence of Lie algebra representations*
> **Input:** *Two faithful representations $\rho_1, \rho_2 \colon \mathcal{L} \to M_n$ of a semisimple (abstract) Lie algebra $\mathcal{L}$. The ρ_i are given by the matrices $\rho_i(b_j)$ for some basis $b_1, \ldots, b_d$ of $\mathcal{L}$, and $\mathcal{L}$ is given by structural constants in the b_i basis (see Appendix A).*
> **Output:** *An outer automorphism $\alpha \in \mathrm{Out}(\mathcal{L})$ such that ρ_1^α is equivalent to ρ_2, or "the two representations are not equivalent up to automorphism."*

Proof. Suppose $\mathcal{L}_1, \mathcal{L}_2 \subseteq M_n$ is an instance of semisimple Lie algebra conjugacy, that is, they are both semisimple matrix Lie algebras. Using techniques given in de Graaf [dG00, §5.11], we can determine if the $\mathcal{L}_i$ are isomorphic as abstract Lie algebras; if not, they are not conjugate as matrix Lie algebras, or if so, we can construct an abstract Lie algebra $\mathcal{L}$ that they are isomorphic to, together with isomorphisms $\rho_i \colon \mathcal{L} \to \mathcal{L}_i$ for $i = 1, 2$. Since $\mathcal{L}_i \subseteq M_n$, the ρ_i are faithful representations of $\mathcal{L}$. We claim that the ρ_i are equivalent up to an outer automorphism of $\mathcal{L}$ if and only if the $\mathcal{L}_i$ are conjugate.

Suppose $\mathcal{L}_2 = g\mathcal{L}_1 g^{-1}$ for some invertible matrix g. Let $c_g \colon M_n \to M_n$ be defined by $c_g(X) = gXg^{-1}$. Then $\alpha = \rho_2^{-1} \circ c_g \circ \rho_1$ is a map from $\mathcal{L}$ to $\mathcal{L}$. Since the ρ_i are isomorphisms, and $c_g|_{\mathcal{L}_1} \colon \mathcal{L}_1 \to \mathcal{L}_2$ is an isomorphism, the composition α is an automorphism of $\mathcal{L}$. Then $\rho_2 \circ \alpha = \rho_2 \circ \rho_2^{-1} \circ c_g \circ \rho_1 = c_g \circ \rho_1$, which is by definition equivalent to ρ_1. By the discussion following Lemma A.3, $\rho_2^{\overline{\alpha}}$ is thus equivalent to ρ_1, where $\overline{\alpha}$ is the outer automorphism corresponding to α.

---

[1]This lemma is essentially present in de Graaf's book [dG00], especially the content leading up to the discussion at the end of his Section 8.5. However, de Graaf's discussion is presented in terms of weights and the choice of Cartan subalgebra, whereas the aspect we wish to highlight requires no mention of these topics, and can be explained by completely elementary means.

Conversely, suppose $\rho_1^{\overline{\alpha}}$ is equivalent to ρ_2 for some outer automorphism $\overline{\alpha}$. Let $\alpha \in \mathrm{Aut}(\mathcal{L})$ be a representative of $\overline{\alpha}$; then there is an invertible matrix g such that $\rho_2 = c_g \circ \rho_1 \circ \alpha$. Then we have

$$\mathcal{L}_2 = \mathrm{Im}(\rho_2) = \mathrm{Im}(c_g \circ \rho_1 \circ \alpha) = c_g(\mathrm{Im}(\rho_1 \circ \alpha)).$$

Since α is an automorphism it is onto, so $\mathrm{Im}(\rho_1 \circ \alpha) = \mathrm{Im}(\rho_1) = \mathcal{L}_1$, and we have $\mathcal{L}_2 = g\mathcal{L}_1 g^{-1}$.

The preceding argument gives a reduction from semisimple Lie algebra conjugacy to the outer equivalence of Lie algebra representations. The reduction in the other direction is as follows: suppose $\mathcal{L}$ is a semisimple Lie algebra and $\rho_1, \rho_2 \colon \mathcal{L} \to M_n$ are two faithful representations. We reduce this to the instance of semisimple Lie algebra conjugacy given by $\mathcal{L}_i = \mathrm{Im}(\rho_i)$ $(i = 1, 2)$. The proof that this is a reduction is identical to the proof above. $\qquad\square$

Lemma 3.3. *Outer equivalence of Lie algebra representations reduces to the following problem:*

> **Problem:** Problem A
> **Input:** *Two $r \times s$ integer matrices M_1, M_2; a partition of the columns into consecutive ranges $[1, \ldots, k_1], [k_1 + 1, \ldots, k_1 + k_2], \ldots [k_1 + \cdots + k_{t-1} + 1, \ldots, s]$; for each range, a group G_ℓ acting on the integers appearing in the corresponding columns, where each G_ℓ is abstractly isomorphic to one of: 1, S_2, or S_3.*
> **Output:** *A permutation $\pi \in S_r$, a permutation $\sigma \in S_{k_1} \times S_{k_2} \times \cdots \times S_{k_t}$, and for each column an element g_j in the group G_ℓ associated to that column range, such that for all i, j, $M_1(i, j) = g_j(M_2(\pi(i), \sigma(j)))$, or "the matrices are not equivalent." In other words, after applying π to the rows, σ to the columns, and each g_j to the values of the entries in the j-th column, M_1 and M_2 become equal.*

Proof. Let $\mathcal{L}$ be a semisimple Lie algebra, and let $\rho_1, \rho_2 \colon \mathcal{L} \to M_n$ be two faithful representations of $\mathcal{L}$. Compute the direct sum decomposition of $\mathcal{L}$; suppose it is $\mathcal{L} = \mathcal{L}_{1,1} \oplus \cdots \oplus \mathcal{L}_{1,k_1} \oplus \mathcal{L}_{2,1} \oplus \cdots \oplus \mathcal{L}_{2,k_2} \oplus \cdots \oplus \mathcal{L}_{t,k_t}$ where each $\mathcal{L}_{i,j}$ is a simple summand of $\mathcal{L}$, and the $\mathcal{L}_{i,j}$ are grouped by isomorphism type, so that $\mathcal{L}_{i_1,j_1}$ and $\mathcal{L}_{i_2,j_2}$ are isomorphic if and only if $i_1 = i_2$. For each i, let $\mathcal{L}_i$ be a simple Lie algebra isomorphic to $\mathcal{L}_{i,j}$ for all j.

To each ρ_i we will associate a matrix M_i, as well as the other data necessary for Problem A. The columns correspond to the direct summands $\mathcal{L}_{i,j}$, and the column partition is along the isomorphism types of the summands.

Next, we define the permutation groups G_ℓ. To each simple type $\mathcal{L}_\ell$, we fix once and for all an encoding of its representations as integers; both the encoding and decoding should be polynomial-time. That this can be done follows from the standard description of the representations of the simple Lie algebras. The integer 0 will always stand for the (trivial) zero representation. The permutation action of $\mathrm{Out}(\mathcal{L}_\ell)$ on the representations of $\mathcal{L}_\ell$, encoded as integers, can be easily computed, as follows. Given $\overline{\alpha} \in \mathrm{Out}(\mathcal{L}_\ell)$ and an integer, convert it to the corresponding representation as above. This representation is a linear map $\mathcal{L}_\ell \to M_n$ for some n. Pre-compose this map with a representative $\alpha \in \mathrm{Aut}(\mathcal{L}_\ell)$ of $\overline{\alpha}$; this can be done because the outer automorphisms of all simple Lie algebras are known explicitly and are easy to compute. For example, the unique outer automorphisms of $\mathfrak{sl}_n$, the trace zero matrices, is given by the map $A \mapsto -A^T$. The outer automorphism groups of simple Lie algebras are all trivial, S_2, or S_3. Finally, convert this new, "twisted-by-α" representation back to an integer. The group G_ℓ associated to the ℓ-th isomorphism type ($=\ell$-th column grouping) is then $\mathrm{Out}(\mathcal{L}_\ell)$, and the action on the integers is the action described above.

Finally, we describe the rows and the entries of the matrices M_i. Decompose the representations ρ_i into their direct sum decompositions $\rho_i = \rho_{i,1} \oplus \cdots \oplus \rho_{i,r}$, where each $\rho_{i,r}$ is an irreducible representation of $\mathcal{L}$. Corollary A.2 says that this can be done in polynomial time. The q-th row of M_i corresponds to the irreducible representations $\rho_{i,q}$. An irreducible representation of a direct sum of Lie algebras is completely specified by its restriction to each summand. Hence, the representation $\rho_{i,q}$ is specified by a representation of each summand $\mathcal{L}_{\cdot,\cdot}$, that is, an integer in each column.

Since the outer automorphism group of $\mathcal{L}$ is $\prod_{i=1}^{t} \text{Out}(\mathcal{L}_i) \wr S_{k_i} = \left(\prod_{i=1}^{t} \text{Out}(\mathcal{L}_i)^{k_i} \right) \rtimes \left(\prod_{i=1}^{t} S_{k_i} \right)$, the representations ρ_1, ρ_2 are equivalent up to an outer automorphism if and only if there is a permutation of the columns (=direct summands of the Lie algebra), for each column an element $g_\ell \in G_\ell$ (=an outer automorphism of each direct summand), and a permutation of the rows (=irreducible constituents of ρ_i) that will make M_1 equal to M_2. Conversely, any such equivalence of M_1 and M_2 according to Problem A corresponds to an outer automorphism of $\mathcal{L}$ that makes ρ_1 and ρ_2 equivalent. $\qquad\square$

Lemma 3.4. *Problem A reduces to graph isomorphism.*

Proof. This is a fun exercise we invite the reader to try for him- or herself. We include the details in Appendix B. $\qquad\square$

Lemma 3.5. *Graph isomorphism reduces to semisimple Lie algebra conjugacy.*

Proof. Let (G_1, G_2) be an instance of graph isomorphism, and let D_i be the 0-1 incidence matrix of G_i, where the rows correspond to edges and the columns correspond to vertices. The G_i are isomorphic if and only if there is a permutation of the rows and the columns that makes the D_i equal. Then (D_1, D_2) is an instance of Problem A where the column partition is trivial, and the column groups G_ℓ are also trivial. We show how to reduce such an instance of Problem A to outer equivalence of Lie algebra representations, and hence to semisimple Lie algebra conjugacy.

Given an instance of Problem A as above—in particular, it only contains the entries 0 and 1, it contains exactly two non-zero entries per row, every column contains a non-zero entry, and the column partition and column groups are all trivial—let $\mathcal{L} = \mathfrak{sl}_2^{\oplus n}$, where n is the number of vertices of the G_i (=columns of the matrices). Let a 1 in the matrix D_i correspond to the adjoint representation of $\mathfrak{sl}_2$ (the Lie algebra of 2×2 trace zero matrices), which is faithful and has dimension 3. Then, by reversing the reduction in Lemma 3.3, we get an instance of outer equivalence of Lie algebra representations.

Since each column contains a non-zero entry, these representations are faithful. Since each row contains exacty two 1's, the corresponding irreducible representation has dimension $3^2 = 9$, hence the representations we get are matrices of dimension $9m \times 9m$, where m is the number of edges of G_i. Since $\mathfrak{sl}_2^{\oplus n}$ is generated by $3n$ elements, the representations can be specified by $3n \times (9m)^2$ numbers, which is polynomial in the size of the original graphs.

Finally, although $\mathfrak{sl}_2$ has an outer automorphism, this outer automorphism acts trivially on the representations of $\mathfrak{sl}_2$, so the corresponding column groups are trivial, as desired. $\qquad\square$

Theorem 3.6. *Conjugacy of semisimple Lie algebras of $n \times n$ matrices can be solved in polynomial time, when the Lie algebras consist of $O(\log n)$ simple direct summands.*

Proof. If there are only $O(\log n / \log \log n)$ simple summands, then an elementary brute-force approach to Problem A works in $poly(n)$ time, since the number of outer automorphisms is $poly(n)$. However, when there are $O(\log n)$ simple summands, the number of outer automorphisms is $n^{O(\log n)}$, so we instead use a more sophisticated approach to twisted code equivalence, due to Babai, Codenotti, and Qiao [BCQ11] (cf. [Cod11, Theorem 4.2.1]). Problem A is in fact a special case of twisted code equivalence with multiplicities, in which each row corresponds to a codeword. On the instance of Problem A corresponding to semisimple Lie algebras with $O(\log n)$ simple summands, their algorithm runs in $poly(n)$ time. Translating between their terminology and ours, the size of the code is the number of rows of the M_i, which is the number irreducible representations of the $\mathcal{L}_i$, which is at most n, the size of the original matrices. Furthermore, the column groups G_i all have bounded size. These two facts together imply that their algorithm runs in $poly(n)$ time. $\qquad\square$

Theorem 3.7. *Conjugacy of semisimple Lie algebras of $n \times n$ matrices can be solved in polynomial time, when the Lie algebras consist of $O(\log n / \log \log n)$ irreducible representations, and unboundedly many simple direct summands, at most $O(\log(n))$ of which have nontrivial outer automorphism actions on their representations.*

In Appendix A.5 we list the simple Lie algebras and their outer automorphism groups, and mention which have trivial actions on their representations. Three of the four infinite families of simple Lie algebras have this property, as well as four of the five exceptional simple Lie algebras.

Proof. In this case, the M_i in the instance of Problem A have only $f(n) \leq O(\log n / \log \log n)$ rows. Although the size of the automorphism group may be more than polynomial, there are only polynomially many row permutations, so we only have to handle the outer automorphisms in each column exhaustively, and not the permutations between the columns. Specifically, try each combination of outer automorphisms of each column; since there are at most $O(\log n)$ columns with nontrivial outer automorphisms, and the outer automorphism group of a simple Lie algebra has size at most 6, there are only $poly(n)$ possibilities. For each such possibility, try each of the $poly(n)$ many permutations of the rows, and for each check whether the set of columns of M_1 is equal to the set of columns of M_2.   □

4. Abelian plus semisimple (i. e., completely reducible)

In this section, we describe how the algorithms and reductions for the abelian diagonalizable and semisimple cases fit into a single general framework and can be combined to handle the case of a direct sum of an abelian diagonalizable matrix Lie algebra with a semisimple matrix Lie algebra. This class of matrix Lie algebras is exactly the class of *completely reducible* matrix Lie algebras. In the case of semisimple Lie algebras we used heavily the fact that all representations of semisimple Lie algebras can be written as a direct sum of irreducible representations (see Appendix A.4). The class we study in this section is the largest class of Lie algebras with this property (cf. Theorem A.1).

Lemma 4.1. *Lemma 3.2 applies to the class of abelian Lie algebras and the class of completely reducible matrix Lie algebras.*

Proof. The proof of Lemma 3.2 only required two ingredients: that the isomorphism problem for abstract Lie algebras of the class under consideration be efficiently solvable, and that twisting a representation by an inner automorphism leads to an equivalent representation. Both of these ingredients hold for abelian Lie algebras: two abelian Lie algebras are abstractly isomorphic if and only if they have the same dimension, and abelian Lie algebras have no non-trivial inner automorphisms.

Similarly, a completely reducible matrix Lie algebra $\mathcal{L}$ is a direct sum $\mathcal{A} \oplus \mathcal{S}$ where $\mathcal{A}$ is abelian diagonalizable and $\mathcal{S}$ is semisimple. The isomorphism problem for this class of Lie algebras is solvable in polynomial time. Finally, $\mathrm{Inn}(\mathcal{L}) = \mathrm{Inn}(\mathcal{A}) \times \mathrm{Inn}(\mathcal{S}) \cong \mathrm{Inn}(\mathcal{S})$ since abelian Lie algebras have no non-trivial inner automorphisms. Hence twisting a representation by an inner automorphism leads to an equivalent representation.   □

Although it was not originally phrased this way, we can now see that the algorithms and equivalences for abelian Lie algebra conjugacy in fact follow the same lines as those for semisimple Lie algebra conjugacy. The main difference is that the outer automorphism group of a d-dimensional abelian Lie algebra is the full general linear group GL_d of $d \times d$ invertible matrices—leading to linear code equivalence—whereas the outer automorphism group of a semisimple Lie algebra is close to S_n—leading to graph isomorphism.

Furthermore, we can view Babai's reduction (see [BCGQ11, Theorem 7.1]) from code equivalence as a sort of "list normal form" algorithm for the action of GL_d by automorphisms. Since GL_d acts by change of basis, we would like reduced row echelon form to be a normal form for this action. However, since one may permute the coordinates in the code equivalence problem, computing reduced row echelon form requires first picking the pivots. Babai's algorithm picks these pivots in all $\binom{n}{d}$ possible ways, reduces to row echelon form, and then uses graph isomorphism to handle the permutation action on the remaining coordinates of the code.

Combining these techniques yields:

Theorem 4.2. *Conjugacy of completely reducible matrix Lie algebras with an abelian diagonalizable part of dimension a, s simple direct summands, and r irreducible representation constituents reduces to $\binom{r}{a}$ instances of Problem A of size $r \times s$. In particular, completely reducible matrix Lie algebra conjugacy of $n \times n$ matrices can be solved in $poly(n)$ time under either of the following conditions:*

- *$a = O(\log n)$, $r = O(1)$, s unbounded, and the number of simple summands with non-trivial outer automorphism action is at most $O(\log n)$;*
- *$a = O(1)$, r unbounded, $s = O(\log n)$.*

5. Application to equivalence of polynomials

Corollary 5.1. *Given the Lie algebra of the symmetry group of a polynomial f on n^2 variables, one can determine whether f is linearly equivalent to $\det_n$ in deterministic $poly(n)$ time.*

Remark 5.2. Computing the Lie algebra of the symmetry group of a polynomial is in fact *equivalent* to polynomial identity testing, and hence cannot be derandomized without proving significant lower bounds [KI04]. Kayal [Kay11a, Lemma 26] shows how to compute the Lie algebra of the symmetry group of a polynomial given as a black-box, using the algorithm from [Kay11b] for computing the linear dependincies between a set of polynomials. Kayal [Kay11b] noted that computing such linear dependencies reduces to the search version of polynomial identity testing. The search and decision versions of polynomial identity testing are equivalent for low-degree functions. Conversely, a polynomial is constant if and only if its symmetry group consists of all invertible transformations of the variables. This holds if and only if the Lie algebra of its symmetry group consists of all linear transformations of the variables. Once this has been determined, evaluating the polynomial at any single point will determine whether it is zero or a non-zero constant.

In some sense, we have thus derandomized Kayal's algorithm as far as is possible in the black-box setting without derandomizing polynomial identity testing. Kayal uses randomization at several points, not just in the computation of the Lie algebra of the symmetry group, and we derandomize those using our deterministic algorithm for semisimple Lie algebra conjugacy from Theorem 3.6.

However, even in the dense, non-black-box setting this represents an improvement from $2^{O(n^2)}$ to $2^{O(n \log n)}$. This is essentially optimal, since a generic function that is equivalent to $\det_n$ will include nearly all monomials of degree n in n^2 variables, of which there are $2^{\Theta(n \log n)}$. By the dense setting we mean the setting in which f is given by a list of coefficients of all monomials of degree n in n^2 variables (without loss of generality, f is homogeneous of degree n). Naive derandomization of Kayal's algorithm takes time $2^{O(n^2)}$, since step (ii) of his §6.2.1 guesses a random element of a space of dimension $\Theta(n^2)$. Similarly, testing affine equivalence to the determinant can be solved using quantifier elimination (see, e.g., Basu, Pollack, and Roy [BPR06, Ch. 14]), again in time $2^{O(n^2)}$, because the witness to equivalence is an $n \times n$ matrix together with an n-dimensional vector. However, computing the Lie algebra of the symmetry group of f only requires solving a linear system of n^2 equations in a number of variables equal to the number of monomials possible, which is roughly $\binom{n^2}{n} \le n^{2n} = 2^{O(n \log n)}$.

Proof of Corollary 5.1. The Lie algebra of the symmetry group of $\det_n$ is $\mathfrak{sl}_n \oplus \mathfrak{sl}_n$, which has only two simple factors. By Theorem 3.6 we can test if the Lie algebra of the symmetry group of f is conjugate to that of $\det_n$. If it is, then act on f by the conjugating matrix so that the Lie algebra is now equal to that of $\det_n$. One might expect to then have to check whether $f(X) = f(X^T)$, since this is also part of the symmetry group of $\det_n$, however, this is not necessary: in the case of the determinant, any function whose symmetry group has a Lie algebra conjugate to that of the determinant is in fact linearly equivalent to the determinant. Note that we have combined here all three main steps of Kayal's algorithm into a single reduction to Lie algebra conjugacy: Kayal uses the Lie algebra to reduce to permutational and scaling equivalence, then solves permutational equivalence and scaling equivalence separately. $\qquad\square$

6. Conclusion and future work

Lie algebra conjugacy arises in Geometric Complexity Theory and the affine equivalence problem. We solved Lie algebra conjugacy over $\mathbb{C}$ in polynomial time for several important classes of Lie algebras—namely abelian, semisimple, and completely reducible (=abelian diagonalizable $\oplus$ semisimple)—under various quantitative constraints. We showed that without these quantitative constraints, these cases of Lie algebra conjugacy all become at least as hard as graph isomorphism.

The completely reducible case is not far from the general case, though significant obstacles remain. Levi's Theorem says that every Lie algebra is the semi-direct product of a solvable Lie algebra by a semisimple one; a solvable Lie algebra is an iterated extension of abelian Lie algebras (see Appendix A.7 for definitions). The completely reducible case, which we resolved, restricts the solvable part to be abelian, and restricts the semidirect product to be direct. The complexity of Lie algebra conjugacy in general remains open, but we believe the following is an achievable next target:

Open Question 6.1. What is the complexity of matrix Lie algebra conjugacy for Lie algebras that are *semi*direct products of abelian by semisimple?

For the abelian diagonalizable case, our results hold over any field. But for the semisimple and completely reducible cases, we only worked over $\mathbb{C}$. The representation theory of semisimple Lie algebras changes in positive characteristic or over non-algebraically closed fields.

Open Question 6.2. What is the complexity of Lie algebra conjugacy over algebraically closed fields of positive characteristic? Over $\mathbb{R}$, $\mathbb{Q}$, number fields, or finite fields?

We essentially derandomized Kayal's algorithm for testing equivalence to the determinant, except for a part of the algorithm that is equivalent to polynomial identity testing. It would be nice to know whether there is a way around this, though we suspect there is not:

Open Question 6.3. Show that testing equivalence to the determinant is as hard as polynomial identity testing, or give a deterministic polynomial-time algorithm for it in the black-box setting. In the dense setting, can equivalence to the determinant be tested in time $poly(t)$ where t is the number of non-zero monomials of the input function?

Finally, there are two avenues for futher progress on the affine equivalence problem using Lie algebra conjugacy. First, although GI-hardness may seem to be the "final" word in the short term, in the application to affine equivalence we may be able to avoid GI altogether. Lie algebra conjugacy is most directly useful for testing affine equivalence to symmetry-characterized functions such as the determinant. A function f is *symmetry-characterized* if for any function g, if g has the same symmetries as f—that is, $f(A\mathbf{x}) = f(\mathbf{x})$ implies $g(A\mathbf{x}) = g(\mathbf{x})$—then g is a scalar multiple of f. Not every Lie algebra can arise as the Lie algebra of the symmetries of a symmetry-characterized function. It is possible that the properties of such Lie algebras are strong enough to avoid graph isomorphism. Second, in addition to the Lie algebra of the symmetries of a function, a function may have a finite group of symmetries "sitting on top of" the Lie algebra.

Open Question 6.4. What is the complexity of testing conjugacy of finite groups of symmetries, arising from symmetry-characterized functions?

References

[BCGQ11] László Babai, Paolo Codenotti, Joshua A. Grochow, and Youming Qiao, *Code equivalence and group isomorphism*, ACM-SIAM Symposium on Discrete Algorithmas (SODA11), 2011.

[BCQ11] László Babai, Paolo Codenotti, and Youming Qiao, *Testing isomorphism of groups with no abelian normal subgroups*, 2011, In preparation.

[BPR06] Saugata Basu, Richard Pollack, and Marie-Françoise Roy, *Algorithms in real algebraic geometry*, second ed., Algorithms and Computation in Mathematics, vol. 10, Springer-Verlag, Berlin, 2006.

[Che46] Claude Chevalley, *Theory of Lie Groups. I*, Princeton Mathematical Series, vol. 8, Princeton University Press, Princeton, N. J., 1946.

[Cod11] Paolo Codenotti, *Testing isomorphism of combinatorial and algebraic structures*, Ph.D. thesis, University of Chicago, Chicago, IL, 2011.

[dG00] Willem A. de Graaf, *Lie algebras: theory and algorithms*, North-Holland Mathematical Library, vol. 56, North-Holland Publishing Co., Amsterdam, 2000.

[FH91] William Fulton and Joe Harris, *Representation theory*, Graduate Texts in Mathematics, vol. 129, Springer-Verlag, New York, 1991, A first course, Readings in Mathematics.

[Geo82] Howard Georgi, *Lie algebras in particle physics*, Frontiers in Physics, vol. 54, Benjamin/Cummings Publishing Co. Inc. Advanced Book Program, Reading, Mass., 1982, From isospin to unified theories, With an introduction by Sheldon L. Glashow.

[Hum78] James E. Humphreys, *Introduction to Lie algebras and representation theory*, Graduate Texts in Mathematics, vol. 9, Springer-Verlag, New York, 1978, Second printing, revised.

[Jac62] Nathan Jacobson, *Lie algebras*, Interscience Tracts in Pure and Applied Mathematics, No. 10, Interscience Publishers (a division of John Wiley & Sons), New York-London, 1962.

[Kay11a] Neeraj Kayal, *Affine projections of polynomials*, Tech. Report TR11-061, Electronic Colloquium on Computational Complexity, 2011.

[Kay11b] Neeraj Kayal, *Efficient algorithms for some special cases of the polynomial equivalence problem*, ACM-SIAM Symposium on Discrete Algorithmas (SODA11), 2011.

[KI04] Valentine Kabanets and Russell Impagliazzo, *Derandomizing polynomial identity tests means proving circuit lower bounds*, Comput. Complexity **13** (2004), no. 1-2, 1–46.

[Kna02] Anthony W. Knapp, *Lie groups beyond an introduction*, second ed., Progress in Mathematics, vol. 140, Birkhäuser Boston Inc., Boston, MA, 2002.

[MS01] Ketan D. Mulmuley and Milind Sohoni, *Geometric complexity theory I: an approach to the P vs. NP and related problems*, SIAM J. Comput. **31** (2001), no. 2, 496–526.

[Olv93] Peter J. Olver, *Applications of Lie groups to differential equations*, second ed., Graduate Texts in Mathematics, vol. 107, Springer-Verlag, New York, 1993.

[OV90] A. L. Onishchik and È. B. Vinberg, *Lie groups and algebraic groups*, Springer Series in Soviet Mathematics, Springer-Verlag, Berlin, 1990, Translated from the Russian and with a preface by D. A. Leites.

[PR97] Erez Petrank and Ron M. Roth, *Is code equivalence easy to decide?*, IEEE Transactions on Information Theory **43** (1997), 1602–1604.

[Ste07] Willi-Hans Steeb, *Continuous symmetries, Lie algebras, differential equations and computer algebra*, second ed., World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2007.

Appendix A. Lie algebra background

For the purposes of this paper, we highly recommend the book of de Graaf [dG00]. We summarize the necessary highlights here. For more general background on Lie algebras we recommend any of several standard books [FH91, Jac62, Hum78, Kna02]. Since we are only working over $\mathbb{C}$ for much of this paper, we omit further mention of the field. However, some of the statements and results

below hold only if the characteristic of the field is zero, and some only if the field is furthermore algebraically closed.

A.1. Basic definitions.

A *Lie algebra* is a vector space $\mathcal{L}$ together with a bilinear operation, referred to as the *Lie bracket* and written $[\cdot, \cdot] \colon \mathcal{L} \times \mathcal{L} \to \mathcal{L}$ satisfying:

   (1) Skew-symmetry: $[v_1, v_2] = -[v_2, v_1]$ (or equivalently, $[v, v] = 0$ for all $v \in \mathcal{L}$)
   (2) Bi-linearity: $[\alpha v + \beta w, u] = \alpha[v, u] + \beta[w, u]$, and similarly for the second coordinate.
   (3) The Jacobi identity: $[u, [v, w]] + [w, [u, v]] + [v, [w, u]] = 0$. This is the "Lie algebra" version of associativity, and can be thought of as "the derivative of the associative law."

A *matrix Lie algebra* is a set of matrices where taking $[A, B] := AB - BA$ makes the set into a Lie algebra. In particular, the collection M_n of all $n \times n$ matrices is a matrix Lie algebra.

A *homomorphism* between Lie algebras $\mathcal{L}_1, \mathcal{L}_2$ is a linear map $\rho \colon \mathcal{L}_1 \to \mathcal{L}_2$ that preserves the brackets, that is, where $\rho([u, v]_{\mathcal{L}_1}) = [\rho(u), \rho(v)]_{\mathcal{L}_2}$. An *isomorphism* is a bijective homomorphism; an *automorphism* is an isomorphism of $\mathcal{L}$ with itself.

Note that conjugate matrix Lie algebras are isomorphic as abstract Lie algebras, since $g[M_1, M_2]g^{-1} = [gM_1g^{-1}, gM_2g^{-1}]$, that is, conjugation by g is a Lie algebra homomorphism whose inverse is conjugation by g^{-1}.

A.2. Describing Lie algebras as input to algorithms.

An abstract Lie algebra is specified in an algorithm by giving a basis for it as a vector space, say $v_1, \ldots, v_d$, and its *structure constants* $c_{ij}^{(k)}$:

$$[v_i, v_j] = \sum_{k=1}^{n} c_{ij}^{(k)} v_k.$$

Because of the bilinearity of the bracket, the structure constants are enough to determine the value of the bracket on any elements of the Lie algebra: $[\sum \alpha_i v_i, \sum \beta_j v_j] = \sum_{ijk} \alpha_i \beta_j c_{ij}^{(k)} v_k$. Each of the axioms of a Lie algebra translates into a condition on the structure constants, for example, skew-symmetry is equivalent to $c_{ij}^{(k)} = -c_{ji}^{(k)}$ for all i, j, k.

A.3. Structure theory of Lie algebras.

Given any two Lie algebras $\mathcal{L}_1, \mathcal{L}_2$, their *direct sum* is the Lie algebra $\mathcal{L}_1 \oplus \mathcal{L}_2$ whose underlying vector space is the direct sum of the underlying vector spaces of the $\mathcal{L}_i$. The bracket $[v_1, v_2]$ for any elements $v_1 \in \mathcal{L}_1$ and $v_2 \in \mathcal{L}_2$ is defined to be zero.

An *ideal* in a Lie algebra is a subspace $I \subseteq \mathcal{L}$ such that $[u, v] \in I$ for any $u \in \mathcal{L}$ and $v \in I$. Ideals are the Lie-algebraic analogue of normal subgroups of groups. Given any ideal, one can form the quotient Lie algebra $\mathcal{L}/I$ whose elements are additive cosets of I, that is, of the form $v + I$; conversely, given any homomorphism of Lie algebras its kernel is an ideal.

A Lie algebra is *abelian* if $[u, v] = 0$ for all $u, v \in \mathcal{L}$. Any vector space can thus be given the structure of an abelian Lie algebra. Every subspace of an abelian Lie algebra is an ideal.

0 is the trivial ideal. An ideal is proper if it is not the whole Lie algebra. In a direct sum $\mathcal{L} = \mathcal{L}_1 \oplus \mathcal{L}_2$, each $\mathcal{L}_i$ is a proper ideal of $\mathcal{L}$. A Lie algebra is *simple* if it contains no proper non-trivial ideals, and is non-abelian. (This last condition excludes, for technical reasons, the 1-dimensional abelian Lie algebra.) A Lie algebra is *semisimple* if it is a direct sum of simple Lie algebras.

Over $\mathbb{C}$, the simple Lie algebras have been completely classified for nearly a century. They fall into four infinite families, referred to as type A_n ($\mathfrak{sl}_n$, consisting of all trace zero $n \times n$ matrices), B_n ($\mathfrak{so}_{2n+1}$, consisting of all $(2n+1) \times (2n+1)$ skew-symmetric matrices $M = -M^T$), C_n ($\mathfrak{sp}_{2n}$ consisting of all $2n \times 2n$ matrices M satisfying $JM = -M^T J$ where $J = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix}$), and D_n ($\mathfrak{so}_{2n}$), and there are five exceptional simple Lie algebras, known as $\mathfrak{e}_6$, $\mathfrak{e}_7$, $\mathfrak{e}_8$, $\mathfrak{f}_4$, and $\mathfrak{g}_2$.

A.4. **Representations.** A *representation of a Lie algebra* $\mathcal{L}$ is a homomorphism $\rho\colon \mathcal{L} \to M_n$ for some n. A representation is *faithful* if this homomorphism is injective. Two representations $\rho_1, \rho_2\colon \mathcal{L} \to M_n$ are *equivalent* if there is an invertible $n \times n$ matrix g such that $\rho_1(v) = g\rho_2(v)g^{-1}$ for all $v \in \mathcal{L}$.

Equivalence of representations is similar to, but not the same as, conjugacy of matrix Lie algebras. Given two representations $\rho_1, \rho_2\colon \mathcal{L} \to M_n$, their images $\mathcal{L}_i := \mathrm{Im}(\rho_i)$ are matrix Lie algebras. The representations ρ_i are equivalent if they are conjugate *as maps*, whereas the matrix Lie algebras forget the maps and only care about their images. In fact, Lemma 3.2 shows that $\mathcal{L}_1$ and $\mathcal{L}_2$ are conjugate matrix Lie algebras if and only if ρ_1 and ρ_2 are equivalent up to an automorphism of $\mathcal{L}$, that is, ρ_1 is equivalent to $\rho_2 \circ \alpha$ for some automorphism $\alpha\colon \mathcal{L} \to \mathcal{L}$. These automorphisms are what cause all the computational difficulties, and allow the equivalences with graph isomorphism and code equivalence.

If $\mathcal{L}$ is specified by a basis and structure constants as above, then a representation $\rho\colon \mathcal{L} \to M_n$ may be specified by giving the k matrices $\rho(v_i)$ for each basis element.

Given two representations $\rho_i\colon \mathcal{L} \to M_{n_i}$ for $i = 1, 2$, their *direct sum* $\rho_1 \oplus \rho_2\colon \mathcal{L} \to M_{n_1+n_2}$ is defined by the block-matrix:

$$(\rho_1 \oplus \rho_2)(v) = \begin{pmatrix} \rho_1(v) & \\ & \rho_2(v) \end{pmatrix}.$$

A representation is called *decomposable* if it is (equivalent to) a non-trivial direct sum; otherwise it is called *indecomposable*.

The set M_n of $n \times n$ matrices acts on the vector space $\mathbb{F}^n$ by the usual matrix-vector multiplication. Given a subset $S \subseteq M_n$, if $V \subseteq \mathbb{F}^n$ is a subspace such that $S \cdot V \subseteq V$, then V is called an *S-invariant subspace*. The 0 subspace and the whole space $\mathbb{F}^n$ are S-invariant for any S.

A representation $\rho\colon \mathcal{L} \to M_n$ is called *irreducible* if 0 and $\mathbb{F}^n$ are the only $\mathrm{Im}(\rho)$-invariant subspaces. Otherwise a representation is called *reducible*. Note that a decomposable representation is reducible, but the converse need not be true, as illustrated by the example:

$$\{ \begin{pmatrix} 1 & x \\ & 1 \end{pmatrix} : x \in \mathbb{F} \}.$$

A representation is *completely reducible* if it can be decomposed into a direct sum of irreducible representations. Every representation can be decomposed into indecomposable representations; in a completely reducible representation these indecomposables must also be irreducible.

A matrix Lie algebra $\mathcal{L} \subseteq M_n$, can be viewed as the image of a faithful representation of $\mathcal{L}$, namely, take $\rho\colon \mathcal{L} \to M_n$ to be the inclusion (i.e., identity) map. Via this identification, we also apply the terms (in)decomposable and (ir)reducible to matrix Lie algebras. If $\mathcal{L}$ is a completely reducible matrix Lie algebra, then it is equivalent (conjugate) to a matrix Lie algebra consisting of block-diagonal matrices, where the restriction to each block is irreducible.

Theorem A.1 (see Theorem III.10 on p. 81 of Jacobson [Jac62]). *A matrix Lie algebra $\mathcal{L}$ is completely reducible if and only if $\mathcal{L}$ is isomorphic to the direct sum of an abelian, diagonalizable Lie algebra and a semisimple Lie algebra.*

The proof of this theorem given in Jacobson [Jac62] is algebraic in nature and can be made effective. All that is required is the solution of a few polynomially sized linear systems of equations. In other words, in polynomial time one can find the irreducible direct summands of a completely reducible representation:

Corollary A.2. *Given a completely reducible matrix Lie algebra $\mathcal{L} \subseteq M_n$, one can find in $\mathrm{poly}(n)$ time a matrix g so that $g\mathcal{L}g^{-1}$ is the direct sum of an abelian diagonal Lie algebra and a semisimple Lie algebra, where the semisimple part consists of block-diagonal matrices, each block being irreducible.*

A.5. **Inner and Outer Automorphisms.** The collection of automorphisms of a Lie algebra $\mathcal{L}$ form a group $\mathrm{Aut}(\mathcal{L})$ under composition of maps. Given a Lie algebra $\mathcal{L}$ and $v \in \mathcal{L}$, the Jacobi identity implies that the map $\mathrm{ad}_v \colon \mathcal{L} \to \mathcal{L}$ defined by $\mathrm{ad}_v(u) := [v, u]$ is a homomorphism of Lie algebras. If $\mathrm{ad}_v^k := \mathrm{ad}_v \circ \cdots \circ \mathrm{ad}_v$ is the zero map for k sufficiently large, then $\exp(\mathrm{ad}_v) := I + \mathrm{ad}_v + \frac{1}{2}\mathrm{ad}_v^2 + \cdots + \frac{1}{(k-1)!}\mathrm{ad}_v^{k-1}$ is an automorphism of $\mathcal{L}$. Automorphisms arising in this way are called *inner automorphisms*. The inner automorphisms form a normal subgroup $\mathrm{Inn}(\mathcal{L}) \leq \mathrm{Aut}(\mathcal{L})$. The quotient group $\mathrm{Aut}(\mathcal{L})/\mathrm{Inn}(\mathcal{L})$ is called the *outer automorphism* group and is denoted $\mathrm{Out}(\mathcal{L})$.

The outer automorphism groups of the simple Lie algebras are completely known:

$$
\begin{aligned}
& & \mathrm{Out}(\mathfrak{sl}_n) &= S_2 & \mathrm{Out}(\mathfrak{sp}_{2n}) &= 1 \\
(n \neq 4) \quad & & \mathrm{Out}(\mathfrak{so}_{2n}) &= S_2 & \mathrm{Out}(\mathfrak{so}_{2n+1}) &= 1 \\
& & \mathrm{Out}(\mathfrak{so}_8) &= S_3 & \mathrm{Out}(\mathfrak{e}_7) &= 1 \\
& & \mathrm{Out}(\mathfrak{e}_6) &= S_2 & \mathrm{Out}(\mathfrak{e}_8) &= 1 \\
& & & & \mathrm{Out}(\mathfrak{f}_4) &= 1 \\
& & & & \mathrm{Out}(\mathfrak{g}_2) &= 1
\end{aligned}
$$

The action of $\mathrm{Out}(\mathfrak{sl}_n)$ on the representations of $\mathfrak{sl}_n$ is trivial. The action technically takes a representation to its dual, but for $\mathfrak{sl}_n$, the dual of a representation is equivalent to that representation.

A.6. **Twisting representations by automorphisms.** Given an automorphism $\alpha \colon \mathcal{L} \to \mathcal{L}$ and a representation $\rho \colon \mathcal{L} \to M_n$, we get another representation $\rho \circ \alpha \colon \mathcal{L} \xrightarrow{\alpha} \mathcal{L} \xrightarrow{\rho} M_n$, given by $(\rho \circ \alpha)(v) = \rho(\alpha(v))$. Since α is an automorphism, it is, in particular, onto, so $\mathrm{Im}(\rho \circ \alpha) = \mathrm{Im}(\rho)$. However, $\rho \circ \alpha$ and ρ need not be equivalent as representations, despite having the same image. We call $\rho \circ \alpha$ the *twist* of the representation ρ by the automorphism α.

For semisimple Lie algebras, twisting by inner automorphisms does in fact lead to equivalent representations:

Lemma A.3 (see Lemma 8.5.1 in de Graaf [dG00])**.** *Let $\rho \colon \mathcal{L} \to M_n$ be a representation of a semisimple Lie algebra $\mathcal{L}$ and let α be an inner automorphism of $\mathcal{L}$. Then $\rho \circ \alpha$ is equivalent to ρ.*

Since twisting a representation by an inner automorphism sends it to an equivalent representation, we find that the outer automorphism group $\mathrm{Out}(\mathcal{L})$ acts on the set of representations-up-to-equivalence. If $\alpha \in \mathrm{Out}(\mathcal{L})$, we denote the image of ρ under the action of α by ρ^α. Equivalently, let $\alpha_* \in \mathrm{Aut}(\mathcal{L})$ be a representative of $\alpha \in \mathrm{Out}(\mathcal{L})$; then ρ^α is the equivalence class of $\rho \circ \alpha_*$, and by the lemma, this equivalence class is independent of the choice of representative α_*.

We note that the same result is vacuously true for abelian Lie algebras, since if $\mathcal{L}$ is abelian then it has no non-trivial inner automorphisms. Hence it also holds for Lie algebras that are a direct sum of abelian and semisimple.

A.7. **More structure theory.** Given two ideals $A, B \subseteq \mathcal{L}$, their commutator is defined as $[A, B] := \mathrm{Span}\{[a, b] : a \in A, b \in B\}$; the commutator of two ideals is again an ideal (this is an exercise in the Jacobi identity). The *derived series* of $\mathcal{L}$ is defined as follows: $\mathcal{L}^{(0)} := \mathcal{L}$, $\mathcal{L}^{(i+1)} := [\mathcal{L}^{(i)}, \mathcal{L}^{(i)}]$. $\mathcal{L}^{(1)} = [\mathcal{L}, \mathcal{L}]$ is called the *derived or commutator subalgebra*.

Definition A.4. A Lie algebra $\mathcal{L}$ is *solvable* if the derived series terminates at $\mathcal{L}^{(k)} = 0$ for some k.

Each step in the derived series, $\mathcal{L}^{(i)}/\mathcal{L}^{(i+1)}$ is abelian, so solvable Lie algebras are "iterated extensions of abelian Lie algebras."

The *lower central series* is defined by $\mathcal{L}_0 := \mathcal{L}$ and $\mathcal{L}_{i+1} := [\mathcal{L}, \mathcal{L}_i]$. Note that here we take the commutator of $\mathcal{L}_i$ with the whole of $\mathcal{L}$, rather than just with $\mathcal{L}_i$ (as in the derived series). Hence the lower central series decreases more slowly than the derived series.

Definition A.5. A Lie algebra $\mathcal{L}$ is *nilpotent* if the lower central series terminates at $\mathcal{L}_k = 0$ for some k.

Finally, in order to state the main structural theorems of Lie algebras, we define semidirect products and derivations. A *derivation* on a Lie algebra $\mathcal{L}$ is a linear map $d \colon \mathcal{L} \to \mathcal{L}$ such that $d([u,v]) = [u, d(v)] + [d(u), v]$. Note the similarity with the product rule for differentiation. Since a derivation is a linear map, we may compose two derivations as linear maps; then defining $[d_1, d_2] := d_1 \circ d_2 - d_2 \circ d_1$ makes the collection of derivations of $\mathcal{L}$ into a Lie algebra denoted $\operatorname{Der}(\mathcal{L})$.

Given two Lie algebras $\mathcal{L}_1, \mathcal{L}_2$ and a homomorphism $\varphi \colon \mathcal{L}_2 \to \operatorname{Der}(\mathcal{L}_1)$, we define the semi-direct product $\mathcal{L}_1 \rtimes_\varphi \mathcal{L}_2$ as follows. The underlying vector space is the direct sum of $\mathcal{L}_1$ and $\mathcal{L}_2$. On each of these subspaces, the Lie bracket is defined as it was originally. If $v \in \mathcal{L}_1$ and $d \in \mathcal{L}_2$ we define

$$[v, d] := d(v).$$

Extending by linearity and skew-symmetry, we find

$$[v_1 + d_1, v_2 + d_2] = [v_1, v_2] + d_2(v_1) - d_1(v_2) + [d_1, d_2]$$

where $v_i \in \mathcal{L}_1$ and $d_i \in \mathcal{L}_2$.

The following two theorems are quite strong structural theorems. For example, nothing even close to these holds in the case of finite groups, despite the similarity in the definitions of all the notions (nilpotent, solvable, semidirect product).

Theorem A.6 (Levi's Theorem, cf. §III.9, p. 91 of Jacobson [Jac62])**.** *Every Lie algebra is the semidirect product of a solvable Lie algebra by a semisimple one. (That is, the semisimple one acts as derivations on the solvable one.)*

Theorem A.7 (see Corollary II.7.1 on p. 51 of Jacobson [Jac62])**.** *A Lie algebra is solvable if and only if its derived subalgebra is nilpotent.*

Remark A.8. Since solvable Lie algebras are iterated extensions of abelian ones (see above), and considering Theorem A.6, we may say that abelian and simple Lie algebras form the "building blocks" of all Lie algebras.

Appendix B. Reduction from Problem A to graph isomorphism

Proof of Lemma 3.4. First, if the permutation groups G_ℓ are all trivial, then we can take each M_i as the bipartite adjacency matrix of a vertex-colored and edge-colored bipartite graph. The vertices corresponding to the columns are colored according to their part in the column partition; we refer to these vertices as column-vertices. The edges are colored by the integer entries of each M_i. It is clear that the M_i are equivalent if and only if the corresponding vertex- and edge-colored bipartite graphs are bipartite-color-isomorphic, that is, isomorphic by an isomorphism which preserves the two parts of the bipartition and preserves each color class of vertices and each color class of edges.

To handle the permutation groups G_i we make one additional step in the reduction. Since there is one G_i for each column i, we must encode its action on the edge-labels incident to the column-vertex i. To do this we add a "color palette" gadget for each column-vertex, which will encode both the edge-labels, as well as enforcing the action of G_i on these labels. That is, the color palette will be such that the way automorphisms of the resulting graph act on the encoding of the edge-labels is exactly the same as G_i acts on them.

To encode the edge-labels with the color palette, we divide each edge by a new vertex, and attach this new vertex to the vertex of the color palette which encodes the appropriate edge color. We only need color palettes capable of encoding permutation actions of the trivial group, S_2, and S_3.

G_i *trivial.* If some G_i is trivial, the corresponding color palette is simply a line of vertices with a marked vertex at the end. The marked vertex prevents reversing the order of the line, and the different vertices in the line encode the different edge labels on the edges incident to column-vertex i.

$G_i \cong S_2$. S_2 has two possible orbit types (=transitive actions): a single fixed point, or an orbit of size two. The color palette is the disjoint union of two graphs corresponding to the two possible

orbit types. Each of these graphs has its own marked vertex at the end. One of these two graphs is simply a line as in the previous case: the vertices of this line correspond to those edge-labels that are fixed by the action of S_2. The other graph is the disjoint union of two lines, each of which is joined at the end to the marked vertex. The action of S_2 swaps the i-th vertex of one of these lines with the i-th vertex of the other. This enforces that the action of the edge group G_i either swaps all of the edge labels (that is, via the nontrivial element of S_2) or none of them.

For the sake of the next case, it is useful to think of this color palette as gluing together in a line multiple copies of the "color gadget" consisting of two disconnected vertices.

$G_i \cong S_3$. S_3 has four orbit types: 1) the trivial action, 2) the action on two points by which odd permutations swap the points and even permutations fix them, 3) the natural action of S_3 on three points, and 4) the regular action of S_3 on itself (6 points). However, these last three orbit types must be linked, since if an element of S_3 swaps two points according to (2), it must also have some action according to (3) and (4). Thus the color palette in this case is the disjoint union of two palettes: the trivial, line palette as before, and a more complicated palette encoding the actions (2)–(4).

This more complicated palette is given by a "color gadget," multiple copies of which are glued together in a line, as in all the other cases. The color gadget is as follows:
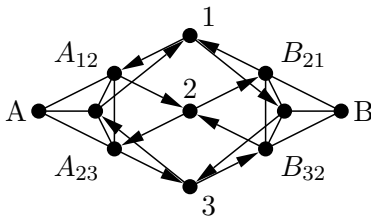
Multiple copies of this color gadget are glued together along three lines, one connecting the "1" vertices, one connecting the "2" vertices, and one connecting the "3" vertices. At one end of these lines, every vertex in the color gadget is connected to a new marked vertex, to prevent the line from being swapped end-to-end.

A set of edge colors corresponding to an orbit of type (2) is encoded by the A and B vertices.

A set of edge colors correspondgin to an orbit of type (3) is encoded by the vertices 1, 2, and 3.

A set of edge colors corresponding to an orbit of type (4) is encoded by the vertices A_{12}, A_{23}, A_{31}, B_{21}, B_{32}, and B_{13}. A_{31} and B_{13} are not labelled in the diagram due to space, but there are directed edges $3 \to A_{31} \to 1$ and $1 \to B_{13} \to 3$.

It remains to show that this color gadget really works as desired. Let us examine the automorphisms of the color gadget. We claim that the automorphism group is S_3, that it acts on the vertices 1, 2, 3 in its natural action (3), it acts on the A_{ij}'s and B_{ji}'s together in its regular action (4), and it acts on A, B in its odd-even action (2).

1, 2, and 3 are the only vertices with in-degree and out-degree 1, so at most they can be swapped amongst each other. Hence the automorphism group is at most S_3. To show the above claim, it suffices to show that the generating set (123) and (12) of S_3 provides automorphisms of the color gadget that act as described.

Consider first (123). It acts on 1, 2, and 3 as described by the cycle notation: $1 \mapsto 2 \mapsto 3 \mapsto 1$. To be an automorphism, it is then forced to send $A_{12} \mapsto A_{23} \mapsto A_{31} \mapsto A_{12}$ and $B_{21} \mapsto B_{32} \mapsto B_{13} \mapsto B_{21}$. Note that (123) cannot possibly swap the A_{ij}'s and the B_{ji}'s, since the directed edges determine an orientation that is preserved by (123). Moreover, its action on these vertices is exactly the action of (123) by right multiplication on S_3 itself. This implies that (123), and hence all the even permutations, fix the vertices A and B.

Next, consider (12). Since (12) reverses the orientation determined by the directed edges, it must swaps the A_{ij}'s and B_{ji}'s, as follows: $A_{12} \leftrightarrow B_{21}$, $A_{23} \leftrightarrow B_{13}$, and $A_{31} \leftrightarrow B_{32}$. This also implies that $A \leftrightarrow B$. Hence odd permutations swap A and B. Finally, the action of (12) on the A_{ij}'s and B_{ji}'s is in accordance with the right regular action of (12) on S_3, compatible with that of (123) above. We can put this together through the correpsondence:

$$
\begin{aligned}
() &\sim A_{12} \\
(123) &\sim A_{23} \\
(132) &\sim A_{31} \\
(12) &\sim B_{21} \\
(13) &\sim B_{32} \\
(23) &\sim B_{13}
\end{aligned}
$$

$\square$

We suspect that these ideas can be extended to show that the general twisted code equivalence problem, as defined in Codenotti [Cod11] Karp-reduces to graph isomorphism.