Adv. appl. Clifford alg. 20 (2010), 121–140

0188-7009/010121-20, *published online* December 11, 2008
DOI 10.1007/s00006-008-0143-2

Advances in Applied Clifford Algebras

Reductions in Computational Complexity Using Clifford Algebras

René Schott and G. Stacey Staples

Abstract. A number of combinatorial problems are treated using properties of abelian null-square-generated and idempotent-generated subalgebras of Clifford algebras. For example, the problem of deciding whether or not a graph contains a Hamiltonian cycle is known to be NP-complete. By considering entries of Λ^k, where Λ is an appropriate nilpotent adjacency matrix, the k-cycles in any finite graph are recovered. Within the algebra context (i.e., considering the number of multiplications performed within the algebra), these problems are reduced to matrix multiplication, which is in complexity class P. The Hamiltonian cycle problem is one of many problems moved from classes NP-complete and ♯P-complete to class P in this context. Other problems considered include the set covering problem, counting the edge-disjoint cycle decompositions of a finite graph, computing the permanent of an arbitrary matrix, computing the girth and circumference of a graph, and finding the longest path in a graph[1].

Mathematics Subject Classification (2000). 68Q15, 05C50, 05C38, 60B99, 81P68.

Keywords. Hamiltonian cycles, travelling salesman problem, longest path, NP-hard, NP-complete, cycle cover, set packing problem, set covering problem, matrix permanent, quantum computing.

1. Introduction

Clifford methods have already been applied to problems in computer vision [1] and automated geometry theorem proving [2]. In work having applications to computer vision, Clifford algebra methods have been employed to estimate points, lines, circles, and spheres from uncertain data while keeping track of the uncertainty [3].

[1] A preliminary version of this paper has been presented at AGACSE 2008 (3rd International Conference on Applied Geometric Algebras in Computer Science and Engineering).

Extending Clifford-algebraic methods to graph theory (cf. [4], [5], [6], [7]) opens the door to applications in theoretical computer science, symbolic dynamics, and coding theory. The methods can be applied to a graph-theoretic construction of multiple stochastic integrals from which stochastic integrals are recovered from the limit in mean of a sequence of Berezin integrals in an ascending chain of Clifford algebras [8].

Clifford algebras have well-known connections with quantum physics and quantum probability [9], [10], [11]. Aerts and Czachor have shown that quantum-like computations can be performed within Clifford algebras without the associated problem of noise and need for error-correction [12].

While Clifford algebra computations can be performed on general purpose processors through the use of software libraries like CLU [13], GluCat [14], GAIGEN [15], and the Maple package CLIFFORD [16], direct hardware implementations of data types and operators is the best way to exploit the computational power of Clifford algebras. To this end, a number of hardware implementations have been developed.

Perhaps the first such hardware implementation was a Clifford co-processor design developed by Perwass, Gebken, and Sommer [17]. Implemented on a Field Programmable Gate Array, the design is scalable in both the dimension of the Clifford algebra and the bit width of the numerical factors.

To our knowledge, the second hardware design was the color edge detection hardware developed by Mishra and Wilson [18], [19]. This focus of their work was the introduction of a hardware architecture for applications involving image processing.

More recently, Gentile, Segreto, Sorbello, Vassallo, Vitabile, and Vullo have developed a parallel embedded coprocessing core that directly supports Clifford algebra operators (cf. [20], [21], [22]). The prototype was implemented on a Field Programmable Gate Array and initial tests showed a 4× speedup for Clifford products over the analogous operations in GAIGEN.

Given a computing architecture based on Clifford algebras, the natural context for determining an algorithm's time complexity is in terms of the number of geometric (Clifford) operations required. This paper assumes the existence of such a processor and examines a number of combinatorial problems known to be of NP time complexity.

For example, the problem of determining whether or not a graph contains a Hamiltonian cycle is known to be NP-complete. By considering entries of Λ^k, where Λ is an appropriate nilpotent adjacency matrix associated with a finite graph on n vertices, the k-cycles in the graph are recovered.

The nilpotent adjacency matrix of a graph on n vertices is defined using elements of an abelian algebra generated by the collection $\{\zeta_i\}$, $1 \leq i \leq n$ satisfying ${\zeta_i}^2 = 0$. In terms of the number of multiplications performed within the algebra, the cycle enumeration problem is reduced to matrix multiplication. While the

algebras used here are not Clifford algebras themselves, they are constructed within Clifford algebras of appropriate signature.

1.1. Notational Preliminaries

Given positive integer n, the n-set $\{1, 2, \ldots, n\}$ will be denoted by $[n]$. The *power set* of the n-set is the set of all subsets of $[n]$ and is denoted by $2^{[n]}$.

Definition 1.1 (Clifford algebra of signature (p, q)). For fixed $n \geq 1$, the 2^n-dimensional algebra $\mathcal{C}\ell_{p,q}$ $(p+q = n)$ is defined as the associative algebra generated by the collection $\{\mathbf{e}_i\}$ $(1 \leq i \leq n)$ along with the unit scalar $\mathbf{e}_0 = \mathbf{e}_\emptyset = 1 \in \mathbb{R}$, subject to the following multiplication rules:

$$\mathbf{e}_i \, \mathbf{e}_j = -\mathbf{e}_j \, \mathbf{e}_i \text{ for } i \neq j, \tag{1.1}$$

$$\mathbf{e}_i{}^2 = \begin{cases} 1 & 1 \leq i \leq p, \text{ and} \\ -1 & p+1 \leq i \leq n. \end{cases} \tag{1.2}$$

Products are multi-indexed by subsets of $[n] = \{1, \ldots, n\}$ in canonical (lexicographical) order according to

$$\mathbf{e}_{\underline{i}} = \prod_{\iota \in \underline{i}} \mathbf{e}_\iota, \tag{1.3}$$

where $\underline{i}$ is an element of the power set $2^{[n]}$. When the generating vectors $\{\mathbf{e}_\iota\}$ are orthonormal, $\mathbf{e}_{\underline{i}}$ is called a *blade*.

Define $f_i = (\mathbf{e}_i - \mathbf{e}_{2n+i}) \in \mathcal{C}\ell_{2n,2n}$ for each $1 \leq i \leq 2n$. Then letting $\zeta_i = f_{2i-1} \, f_{2i}$ for $1 \leq i \leq n$, the following useful algebra is obtained.

Definition 1.2. Let $\mathcal{C}\ell_n{}^{\text{nil}}$ denote the real abelian algebra generated by the collection $\{\zeta_i\}$ $(1 \leq i \leq n)$ along with the scalar $1 = \zeta_0$ subject to the following multiplication rules:

$$\zeta_i \, \zeta_j = \zeta_j \, \zeta_i \quad \text{for } i \neq j, \text{ and} \tag{1.4}$$

$$\zeta_i{}^2 = 0 \quad \text{for } 1 \leq i \leq n. \tag{1.5}$$

It is evident that a general element $u \in \mathcal{C}\ell_n{}^{\text{nil}}$ can be expanded as

$$u = \sum_{\underline{i} \in 2^{[n]}} u_{\underline{i}} \, \zeta_{\underline{i}}, \tag{1.6}$$

where $\underline{i} \in 2^{[n]}$ is a subset of $[n] = \{1, 2, \ldots, n\}$ used as a multi-index, $u_{\underline{i}} \in \mathbb{R}$, and $\zeta_{\underline{i}} = \prod_{\iota \in \underline{i}} \zeta_\iota$.

It is worth noting that the algebra $\mathcal{C}\ell_n{}^{\text{nil}}$ can be constructed in a number of ways. For example, recalling the Clifford algebra $\mathcal{C}\ell_{p,q,r}$ as defined by Porteous [23], in which r of the vectors square to zero, one could define $\zeta_i = \mathbf{e}_{2i-1}\mathbf{e}_{2i} \in \mathcal{C}\ell_{0,0,2n}$ for each $1 \leq i \leq n$. Equivalently, one could consider pairwise disjoint bivectors in the Grassmann algebra.

Letting $\varepsilon_i = \dfrac{1}{2}\left(1 + \mathbf{e}_i\mathbf{e}_{n+i}\right) \in \mathcal{C}\ell_{n,n}$ for each $1 \leq i \leq n$ gives the following algebra.

Definition 1.3. Let ${\mathcal{C}\ell_n}^{\text{idem}}$ denote the real abelian algebra generated by the collection $\{\varepsilon_i\}$ $(1 \leq i \leq n)$ along with the scalar $1 = \varepsilon_0$ subject to the following multiplication rules:

$$\varepsilon_i\,\varepsilon_j = \varepsilon_j\,\varepsilon_i \quad \text{for } i \neq j, \text{ and} \tag{1.7}$$

$${\varepsilon_i}^2 = \varepsilon_i \quad \text{for } 1 \leq i \leq n. \tag{1.8}$$

An element $\beta \in {\mathcal{C}\ell_n}^{\text{idem}}$ can also be expanded as in (1.6); that is,

$$\beta = \sum_{\underline{i} \in 2^{[n]}} \beta_{\underline{i}}\,\varepsilon_{\underline{i}}. \tag{1.9}$$

Both algebras admit an inner product of the form

$$\left\langle \sum_{\underline{i} \in 2^{[n]}} u_{\underline{i}}\,\zeta_{\underline{i}},\ \sum_{\underline{j} \in 2^{[n]}} v_{\underline{j}}\,\zeta_{\underline{j}} \right\rangle = \sum_{\underline{\ell} \in 2^{[n]}} u_\ell\,v_\ell. \tag{1.10}$$

The *grade-k part* of $u \in {\mathcal{C}\ell_n}^{\text{nil}}$ will be defined by

$$\langle u \rangle_k = \sum_{\substack{\underline{i} \in 2^{[n]} \\ |\underline{i}| = k}} u_{\underline{i}}\,\zeta_{\underline{i}}. \tag{1.11}$$

Here $|\underline{i}|$ denotes the number of elements in the multi-index $\underline{i}$.

Letting u denote an arbitrary element of ${\mathcal{C}\ell_n}^{\text{nil}}$, the *scalar sum* of coefficients will be denoted by

$$\langle\langle u \rangle\rangle = \sum_{\underline{i} \in 2^{[n]}} \left\langle u, \zeta_{\underline{i}} \right\rangle = \sum_{\underline{i} \in 2^{[n]}} u_{\underline{i}}. \tag{1.12}$$

The definitions of scalar sum and grade-k part extend naturally to ${\mathcal{C}\ell_n}^{\text{idem}}$.

A number of norms can be defined on ${\mathcal{C}\ell_n}^{\text{nil}}$. One that will be used later is the *infinity norm*, defined by

$$\left\| \sum_{\underline{i} \in 2^{[n]}} u_{\underline{i}}\,\zeta_{\underline{i}} \right\|_\infty = \max_{\underline{i} \in 2^{[n]}} \left| u_{\underline{i}} \right|. \tag{1.13}$$

Remark 1.4. The algebra $\mathcal{C}\ell_{n,n}$ is canonically isomorphic to the fermion algebra of quantum physics [9].

An algorithm's time complexity is typically determined by counting the number of operations required to process a data set of size n in worst-, average-, and best-case scenarios. The operation of multiplying two integers is typical. Multiplying a pair of integers in classical computing is assumed to require a constant interval of time, independent of the integers. The architecture of a classical computer makes this assumption natural.

The existence of a processor whose registers accommodate storage and manipulation of elements of $\mathcal{C}\ell_n{}^{\diamondsuit}$ *is assumed through the remainder of this paper.*

The $\mathcal{C}\ell$ complexity of an algorithm will be determined by the required number of $\mathcal{C}\ell_n{}^{\diamondsuit}$ operations, or *Clops* required by the algorithm. In other words, multiplying (or adding) a pair of elements $u, v \in \mathcal{C}\ell_n{}^{\diamondsuit}$ will require one Clop, where $\diamondsuit$ can be replaced by either "nil" or "idem."

Evaluation of the infinity norm is another matter. In one possible model of such an evaluation, the scalar coefficients in the expansion of $u \in \mathcal{C}\ell_n{}^{\diamondsuit}$ are first paired off and all pairs are then compared in parallel. In this way, evaluation of the infinity norm has complexity
$O(\log 2^n) = O(n)$.

Let $u \in \mathcal{C}\ell_{p,q}$ where $p + q = n$, and consider the following set of operations.

$$u^{\ddagger} = \left(\sum_{\underline{i} \in 2^{[n]}} u_{\underline{i}} \mathbf{e}_{\underline{i}} \right)^{\ddagger} := \sum_{\underline{i} \in 2^{[n]}} u_{\underline{i}} \, \mathbf{e}_{[n] \setminus \underline{i}} \,, \tag{1.14}$$

$$\iota \left(\sum_{\underline{i} \in 2^{[n]}} u_{\underline{i}} \, \mathbf{e}_{\underline{i}} \right) := \sum_{u_{\underline{i}} > 0} \mathbf{e}_{\underline{i}} \,, \tag{1.15}$$

$$u \boxdot v := \sum_{\underline{i} \in 2^{[n]}} \left\langle u, \mathbf{e}_{\underline{i}} \right\rangle \left\langle v, \mathbf{e}_{\underline{i}} \right\rangle \mathbf{e}_{\underline{i}} = \sum_{\underline{i} \in 2^{[n]}} u_{\underline{i}} \, v_{\underline{i}} \, \mathbf{e}_{\underline{i}} \,, \tag{1.16}$$

$$\langle u \rangle_{\oplus} := u \boxdot \iota(u) \,, \tag{1.17}$$

$$\pi_k \left(\sum_{\underline{i} \in 2^{[n]}} u_{\underline{i}} \, \mathbf{e}_{\underline{i}} \right) := \sum_{\underline{i} \in 2^{[k]}} u_{\underline{i}} \, \mathbf{e}_{\underline{i}} . \tag{1.18}$$

Also, define the element

$$\mathbf{1}_{[n]} := \sum_{\underline{i} \in 2^{[n]}} \mathbf{e}_{\underline{i}} . \tag{1.19}$$

Note that $u^{\ddagger}$ is similar to the Hodge dual, but is not.

Using these operations, an algorithm of $\mathcal{C}\ell$ complexity $O(n)$ can now be written for computing the infinity norm.

procedure InfNorm(u: $u \in \mathcal{C}\ell_{p,q}$)
{Replace all scalar coefficients of u with their absolute values.}
$u := \langle u \rangle_{\oplus} + \langle -u \rangle_{\oplus}$
for $j := 1$ **to** n
begin
{Get multi-vectors $\mathbf{e}_{\underline{i}}$ corresponding to $u_{\underline{i}} \geq u_{[n] \setminus \underline{i}}$}

$x := \mathbf{1}_{[n-j+1]} - \iota \left(\left\langle u^{\ddagger} - u \right\rangle_{\oplus} \right)$
$u := u \boxdot x$

{Now u has only half as many nonzero coefficients. Project down onto subalgebra of dimension 2^{n-j}.}

$$u := \pi_{n-j}(u) + \pi_{n-j}\left(u^{\ddagger}\right)$$

{Note that in each case, either $\mathbf{e}_{\underline{i}}$ or $\mathbf{e}_{[n]\setminus\underline{i}}$ is in the subalgebra, exclusively. Hence, $u_{\underline{i}} \geq u_{[n]\setminus\underline{i}}$ implies $u_{\underline{i}}$ is assigned as the coefficient of the appropriate multivector in the subalgebra.}
end
return$[u]$

Given a processor capable of natively performing the operations defined in (1.14)-(1.18), construction of this algorithm establishes the following results.

Lemma 1.5. *The $\mathcal{C}\ell$ complexity of evaluating the infinity norm in ${\mathcal{C}\ell_n}^{\diamondsuit}$ is $O(n)$.*

Lemma 1.6. *The $\mathcal{C}\ell$ complexity of finding the maximum or minimum real number in a list of size k is $O(\log k)$.*

Proof. Using the algorithm, the complexity of finding the maximum or minimum value in a list of size 2^n is $O(n)$. □

Well-known classical sorting algorithms include bubble, insertion, selection, shell, heap, merge, and quick sorts. These algorithms can be divided into two classes based on their algorithmic complexity: those of complexity $O(n^2)$, which includes the bubble, insertion, selection, and shell sorts; and those of complexity $O(n \log n)$ which includes the heap, merge, and quick sorts. Quick sort has a best-case complexity $O(n \log n)$ and worst-case complexity $O(n^2)$.

The next result follows from Lemma 1.6.

Corollary 1.7. *The $\mathcal{C}\ell$ complexity of sorting a list of size k is $O((\log k)^2)$.*

Proof. Given a list of size 2^n, a sorted list is created by repeatedly locating and removing the largest element of the original list. In light of Lemma 1.6, the complexity is

$$O\left(\sum_{k=1}^{n} k\right) = O\left(\frac{n^2+n}{2}\right) = O(n^2). \tag{1.20}$$

Hence, the result. □

2. Graph Problems

A *graph* $G = (V, E)$ is a collection of vertices V and a set E of unordered pairs of vertices called *edges*. Two vertices $v_i, v_j \in V$ are *adjacent* if there exists an edge $\{v_i, v_j\} \in E$.

Given a graph G, it will sometimes be convenient to use the notation V_G and E_G to denote, respectively, the vertices and the edges of G.

A *k-walk* $(v_0, \ldots, v_k)$ in a graph G is a sequence of vertices in G with *initial vertex* v_0 and *terminal vertex* v_k such that there exists an edge $\{v_j, v_{j+1}\} \in E$ for each $0 \leq j \leq k-1$. Note that a k-walk contains k edges. A *k-path* is a k-walk in which no vertex appears more than once. A *closed k-walk* is a k-walk whose initial vertex is also its terminal vertex. A *k-cycle* ($k \geq 3$) is a closed k-path with $v_0 = v_k$. A *Hamiltonian cycle* is an n-cycle in a graph on n vertices; i.e., it contains V.

The *circumference* of a graph is the length of the longest cycle contained in the graph. The *girth* of a graph is defined as the length of the shortest cycle contained in the graph.

When working with a graph G on n vertices, one often utilizes the *adjacency matrix* A associated with G. If the vertices are labeled $\{1, \ldots, n\}$, one defines A by

$$A_{ij} = \begin{cases} 1 & \text{if } v_i, v_j \text{ are adjacent} \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

The following simple but useful result of this definition, which can also be generalized to directed graphs, is given here without proof.

Proposition 2.1. *Let G be a graph on n vertices with associated adjacency matrix A. Then for any positive integer k, the $(i,j)^{th}$ entry of A^k is the number of k-walks $i \to j$. In particular, the entries along the main diagonal of A^k are the numbers of closed k-walks in G.*

Note that although the adjacency matrix can be used to count walks, it does not provide a method of counting paths and cycles. For that, a "new" type of adjacency matrix is needed.

2.1. Nilpotent Adjacency Matrices

Definition 2.2. Let G be a graph on n vertices, either simple or directed with no multiple edges, and let $\{\zeta_i\}$, $1 \leq i \leq n$ denote the null-square generators of ${\mathcal{C}\ell_n}^{\text{nil}}$. Define the *nilpotent adjacency matrix* associated with G by

$$\Lambda_{ij} = \begin{cases} \zeta_j, & \text{if } \{v_i, v_j\} \in E_G \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

It should be clear that Λ defined over ${\mathcal{C}\ell_n}^{\text{nil}}$ implies Λ^k is the $n \times n$ zero matrix for all $k > n$. Therefore $(I - t\Lambda)^{-1} = \sum_{k=0}^{n} t^k \Lambda^k$ exists as a finite sum, and one can recover

$$\operatorname{tr} \Lambda^k = \operatorname{tr} (I - t\Lambda)^{-1} \Big|_{t^k}. \tag{2.3}$$

In other words, the trace of the coefficient of t^k in the power series expansion of $(I - t\Lambda)^{-1}$ is the trace of Λ^k.

Theorem 2.3. *Let Λ be the nilpotent adjacency matrix of an n-vertex graph G. For any $m \geq 3$ and $1 \leq i \leq n$, summing the coefficients of $(\Lambda^m)_{ii}$ yields the number of m-cycles based at v_i occurring in G.*

Proof. The proof is by induction on m. First it will be shown that matrix entry $(\Lambda^m)_{ij}$ corresponds to a sum of blades indexed by vertex sets on which there exist m-step walks from v_i to v_j that revisit no vertex except possibly v_i itself. When $m = 1$, this is true by definition of Λ.

Now assuming the proposition holds for m and considering the case $m + 1$,

$$\left(\Lambda^{m+1}\right)_{ij} = (\Lambda^m \times \Lambda)_{ij} = \sum_{\ell=1}^{n} (\Lambda^m)_{i\ell}\, \Lambda_{\ell j}. \tag{2.4}$$

Considering a general term of the sum,

$$(\Lambda^m)_{i\ell} = \sum_{m\text{-paths } w_m : v_i \to v_\ell} w_m, \text{ and} \tag{2.5}$$

$$\Lambda_{\ell i} = \sum_{1\text{-paths } w_1 : v_\ell \to v_j} w_1. \tag{2.6}$$

Because the vertices are labeled with the null-square generators of ${\mathcal{C}\ell_n}^{\text{nil}}$, any repeated vertex yields 0.

It should then be clear that terms of the product

$$(\Lambda^m)_{i\ell}\, \Lambda_{\ell j} \tag{2.7}$$

are nonzero if and only if they correspond to $(m + 1)$-*paths* $v_i \to v_\ell \to v_j$, with the possible exception that if $i \neq j$, then v_i could be revisited once. Summing over all vertices v_ℓ gives the sum of all $(m + 1)$-walks based at v_i that revisit no vertex with only the specified exception possible.

Finally, when $i = j$, the null-square generator associated with v_i appears as the last factor in each product over the $(m+1)$-walks. As a result, any walks that revisit v_i in an intermediate step are removed from $(\Lambda^{m+1})_{ii}$, leaving only the $(m + 1)$-cycles based at v_i. □

In an undirected graph, two orientations are possible for each cycle. As a result, each m-cycle is represented with multiplicity two along the diagonal of Λ^m. Throughout the remainder of this paper, two cycles in an undirected graph will be considered the "same" if they differ only by orientation or choice of base point.

Remark 2.4. The nilpotent adjacency matrix associated with a finite graph can be considered a quantum random variable whose m^{th} moment corresponds to the number of m-cycles occurring in the graph [5].

The next two corollaries follow immediately from Theorem 2.3

Corollary 2.5. *Let Λ be the nilpotent adjacency matrix of an n-vertex graph G. Let X_m $(m \geq 3)$, denote the number of m-cycles appearing in the graph G. Then*

$$\langle\langle \text{tr}\,(\Lambda^m)\rangle\rangle = 2m\, X_m. \tag{2.8}$$

Notation. To simplify notation, $\text{tr}\,(\Lambda^m)$ is replaced by τ_m in the remainder of the paper.

Corollary 2.6. *Let Λ be the nilpotent adjacency matrix of an n-vertex graph G. Let H_n denote the number of Hamiltonian cycles appearing in the graph G. Then*

$$\langle\langle \tau_n \rangle\rangle = 2n\, H_n. \tag{2.9}$$

Recall the stated assumption that addition of two arbitrary elements of ${\mathcal{C}\ell_n}^{\mathrm{nil}}$ requires 1 Clop. It follows that computing $\mathrm{tr}(A)$, where A is an $n \times n$ matrix having entries in ${\mathcal{C}\ell_n}^{\mathrm{nil}}$, has $\mathcal{C}\ell$ complexity $O(n)$.

Using the Coppersmith-Winograd algorithm, multiplying two $n \times n$ matrices can be done in $O(n^{2.376})$ time [24]. It is not clear that the same asymptotic speedup can be accomplished for the $\mathcal{C}\ell$ case. However, in the remainder of the paper, β will represent the exponent associated with matrix multiplication. In the worst case, multiplication of $n \times n$ matrices with entries in ${\mathcal{C}\ell_n}^{\mathrm{nil}}$ has $\mathcal{C}\ell$ complexity $O(n^3)$, so that $\beta \leq 3$.

The following corollaries are immediate consequences of Theorem 2.3 and the complexity of matrix multiplication.

Corollary 2.7. *Enumerating the k-cycles in a finite graph on n vertices requires $O(n^\beta \log k)$ Clops.*

Corollary 2.8. *Enumerating the Hamiltonian cycles in a finite graph on n vertices requires $O(n^\beta \log n)$ Clops.*

Corollary 2.9. *Let Λ be the nilpotent adjacency matrix of an n-vertex graph G. Let $X_{m,\ell}$ denote the number of ℓ-tuples of pairwise disjoint m-cycles appearing in the graph G, where $m \geq 3$ and $1 \leq \ell \leq \lfloor n/m \rfloor$. Then*

$$\left\langle\left\langle (\tau_m)^\ell \right\rangle\right\rangle = (2m)^\ell \ell! X_{m,\ell}. \tag{2.10}$$

Proof. Note that $\dfrac{\tau_m}{2m}$ is a sum of nilpotent multivectors associated with m-cycles in the graph. By nilpotency, the nonzero terms of $\left(\dfrac{\tau_m}{2m}\right)^\ell$ represent pairwise disjoint m-cycles, and each term occurs $\ell!$ times. □

Corollary 2.10. *Counting the ℓ-tuples of pairwise disjoint m-cycles in a finite graph on n vertices requires $O(n^\beta \log m)$ Clops.*

Proof. As already seen, computing τ_m requires $O(n^\beta \log m)$ Clops. Computing ${\tau_m}^\ell$ then requires $O(\log \ell)$ additional Clops. Hence, counting ℓ-tuples of m-cycles requires $O(\max\{\log \ell, n^\beta \log m\})$, where ℓ is never larger than n/m. □

Recall that the circumference of a graph is defined as the length of the longest cycle contained in the graph. The following proposition is an immediate consequence of Theorem 2.3.

Proposition 2.11 (Graph circumference). *Let G be a graph on n vertices with nilpotent adjacency matrix Λ. The length of the longest cycle in G is the largest integer k such that*

$$\tau_k \neq 0. \tag{2.11}$$

Corollary 2.12. *Computing the circumference of a graph on n vertices requires $O(n^{\beta+1}\log n)$ Clops.*

Proof. Cycles of length k must be counted for $3 \leq k \leq n$, requiring $O\left(n^{\beta}\log k\right)$ for each $3 \leq k \leq n$. Then,

$$\sum_{k=3}^{n} n^{\beta}\log k = n^{\beta}\log(n!/2) \leq n^{\beta}\log(n^n) = n^{\beta+1}\log n. \tag{2.12}$$

□

Recall that the girth of a graph G is defined as the length of the smallest nontrivial cycle in G.

Corollary 2.13 (Graph girth). *Let G be a graph on n vertices with nilpotent adjacency matrix Λ. The length of the shortest cycle in G is the smallest integer $k \geq 3$ such that*

$$\tau_k \neq 0. \tag{2.13}$$

Corollary 2.14. *Computing the girth of a graph on n vertices requires $O(n^{\beta+1}\log n)$ Clops.*

Proof. The proof is similar to that of Corollary 2.12. □

In the next proposition, $\mathcal{C}$ denotes the diagonal matrix $\mathrm{Diag}(\zeta_1, \ldots, \zeta_n)$. It is used to account for the initial vertices of paths in G.

Proposition 2.15 (Longest path). *Let G be a graph on n vertices with nilpotent adjacency matrix Λ. The length of the longest path in G is the largest integer k such that*

$$\mathcal{C}\Lambda^k \neq \mathbf{0}. \tag{2.14}$$

Here, $\mathbf{0}$ denotes the $n \times n$ zero matrix.

Proof. This is a corollary of Theorem 2.3. Cycles are disregarded by removing the diagonal entries of $\mathcal{C}\Lambda^k$. Left multiplication by the matrix $\mathcal{C}$ is used to "sieve" out k-walks that revisit their initial vertices. □

Corollary 2.16. *Computing the length of the longest path in a graph on n vertices requires $O(n^{\beta}\log n \log n)$ Clops.*

Proof. The maximum possible path length is n. For each $1 \leq k \leq n$, computing $\mathcal{C}\Lambda^k$ requires $O(n^{\beta}\log k + n^2) = O(n^{\beta}\log n)$ Clops. Using binary search then requires testing $O(\log n)$ values of k in Proposition 2.15. □

Consider a directed, edge-weighted graph G on n vertices. When $\{v_i, v_j\}$ is an edge of G, let w_{ij} denote the weight or "cost" of the edge. The goal is to compute the total additive weight of all k-cycles in G.

The infinity norm in the following theorem is the natural extension of that found in (1.13).

Theorem 2.17 (Minimum cost of all k-cycles). *Let G be a finite graph on n vertices with m edges of weights $w_1, w_2, \ldots, w_m$. Let $f : V_G \times V_G \to \mathbb{N}$ be a labeling of the edges of G with natural numbers. Label the vertices of G with the null-square generators of ${\mathcal{C}\ell_n}^{\mathrm{nil}}$ and let edges of G be labeled with $\exp(-w_{ij})\gamma_{f(i,j)}$, where $\{\gamma_i\}$ is the collection of null-square generators of ${\mathcal{C}\ell_{|E_G|}}^{\mathrm{nil}}$. The nilpotent adjacency matrix then has entries in ${\mathcal{C}\ell_n}^{\mathrm{nil}} \otimes {\mathcal{C}\ell_{|E_G|}}^{\mathrm{nil}}$. The minimum cost k-cycle in G has cost*

$$W_{\min} = -\ln\left(\left|\left|\frac{1}{2k}\tau_k\right|\right|_\infty\right). \tag{2.15}$$

Proof. Analogous to the proof of Theorem 2.3, the trace of Λ^k consists of elements of the form

$$\sum_{i=1}^{n} \sum_{k\text{-cycles } \xi \text{ based at } v_i} \exp\left(-\sum_{v_\ell \in \xi} w_\ell\right) \zeta_{\underline{i}(\xi)}\, \gamma_{\underline{j}(\xi)}. \tag{2.16}$$

Here, $\underline{i}(\xi)$ denotes the subset of the n-set that corresponds to the vertices in cycle ξ. Similarly, $\underline{j}(\xi)$ is a subset of $\{1, 2, \ldots, |E_G|\}$ representing the edges contained in cycle ξ. As seen previously, each cycle is represented with multiplicity $2k$ in the trace of Λ^k.

Clearly the maximum coefficient in the expansion of the trace corresponds to the minimum sum of weights in the argument of the exponential function. □

Two more results follow immediately from this theorem.

Corollary 2.18. *The problem of determining the minimum cost associated with a k-cycle in G has $\mathcal{C}\ell$ complexity $O(n^\beta \log k)$.*

Corollary 2.19 (Minimum cost Hamiltonian cycle). *The problem of determining the minimum cost associated with a Hamiltonian cycle in G has $\mathcal{C}\ell$ complexity $O(n^\beta \log n)$.*

2.2. Edge-disjoint Cycle Decompositions of Graphs

Consider the algebra $\mathbb{R}[s_1, \ldots, s_n]$ of polynomials in commutative variables $s_1, \ldots, s_n$. Allowing these polynomials to have coefficients in ${\mathcal{C}\ell_n}^{\mathrm{nil}} \otimes {\mathcal{C}\ell_m}^{\mathrm{nil}}$ generates the abelian algebra ${\mathcal{C}\ell_n}^{\mathrm{nil}} \otimes {\mathcal{C}\ell_m}^{\mathrm{nil}}[s_1, \ldots, s_n]$.

Define the projection $\vartheta : {\mathcal{C}\ell_n}^{\mathrm{nil}} \otimes {\mathcal{C}\ell_m}^{\mathrm{nil}}[s_1, \ldots, s_n] \to {\mathcal{C}\ell_m}^{\mathrm{nil}}[s_1, \ldots, s_n]$ by linear extension of

$$\vartheta\left(\alpha\, \zeta_{\underline{j}}\, \gamma_{\underline{i}}\, s_1^{j_1} \cdots s_n^{j_n}\right) = \alpha\, \gamma_{\underline{i}}\, s_1^{j_1} \cdots s_n^{j_n}, \tag{2.17}$$

and define the evaluation $\langle\langle \cdot \rangle\rangle : {\mathcal{C}\ell_m}^{\mathrm{nil}}[s_1, \ldots, s_n] \to \mathbb{R}[s_1, \ldots, s_n]$ by linear extension of

$$\left\langle\left\langle \alpha\, \gamma_{\underline{i}}\, s_1^{j_1} \cdots s_n^{j_n} \right\rangle\right\rangle = \alpha\, s_1^{j_1} \cdots s_n^{j_n}, \tag{2.18}$$

where $\alpha \in \mathbb{R}$.

The projection ϑ and the evaluation $\langle\langle \cdot \rangle\rangle$ will be assumed to have $\mathcal{C}\ell$ complexity $O(1)$.

Definition 2.20. A finite graph G on n vertices will be said to have a *cycle decomposition* if for some positive integer m there exists a collection of cycles $\{C_i\}_{1\leq i\leq m}$ such that

$$V_G = \bigcup_{i=1}^{m} V_{C_i} \tag{2.19}$$

$$E_G = \bigcup_{i=1}^{m} E_{C_i} \tag{2.20}$$

$$E_{C_i} \cap E_{C_j} = \emptyset \text{ if } i \neq j. \tag{2.21}$$

Here, V_G and E_G denote the sets of vertices and edges of the graph G, respectively. The collection $\{C_i\}_{1\leq i\leq m}$ is called the *cycle decomposition* of G.

Every undirected graph has a trivial cycle decomposition consisting of $|E_G|$ two-cycles. For purposes of this paper, only cycles of length greater than two are considered here.

Theorem 2.21. *Let G be a simple graph on n vertices and $|E|$ edges with nilpotent adjacency matrix over ${\mathcal{C}\ell_n}^{\text{nil}} \otimes {\mathcal{C}\ell_{|E|}}^{\text{nil}}$. Then G has a cycle decomposition $G = C_{j_1} \cup \cdots \cup C_{j_m}$, $\sum_i j_i = |E|$ if and only if the degree-m monomial $\alpha\, s_{j_1} \cdots s_{j_m}$ is a term in the expansion of*

$$\left\langle \left\langle \left\langle \left(\vartheta \left(\sum_{k=3}^{n} s_k \frac{\tau_k}{2k} \right) \right)^m \right\rangle_{|E|} \right\rangle \right\rangle \tag{2.22}$$

where $\dfrac{\alpha}{m!} \in \mathbb{N}$ indicates the multiplicity *of the decomposition.*

Proof. Begin by letting Λ be the edge-labeled nilpotent adjacency matrix of an n-vertex graph $G = (V, E)$. From Theorem 2.3, it follows that for any $k \geq 3$ and $1 \leq i \leq n$, summing the coefficients of $(\Lambda^k)_{ii}$ yields the number of k-cycles based at v_i occurring in G.

Hence, $\dfrac{\tau_k}{2k}$ corresponds to the collection of k-cycles in G in one-to-one correspondence.

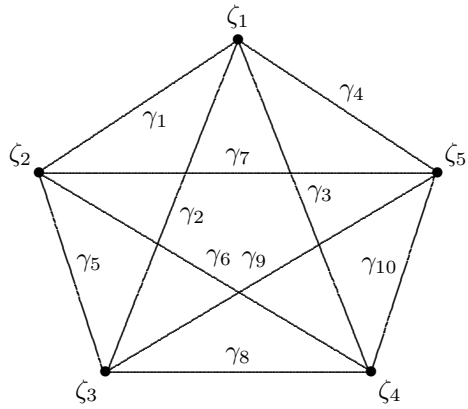
It is now clear that $\displaystyle\sum_{k=3}^{n} s_k \frac{\tau_k}{2k}$ denotes the algebraic sum of all the cycles contained in G. Projecting down onto ${\mathcal{C}\ell_{|E|}}^{\text{nil}}$ leaves each k-cycle represented by a grade-k multivector $\gamma_{\underline{i}}$, corresponding to the edges comprising the cycle.

By nilpotency, the nonzero terms of $\vartheta \left(\displaystyle\sum_{k=3}^{n} s_k \frac{\tau_k}{2k} \right)^m$ represent the collection of all m-ensembles of pairwise edge-disjoint cycles, and each term occurs $m!$ times in the expansion. By considering only those terms of grade $|E|$ in ${\mathcal{C}\ell_{|E|}}^{\text{nil}}$, one ensures that only cycle decompositions of G are obtained since all edges are represented. □

Corollary 2.22. *Let G be a simple graph on n vertices and $|E|$ edges with nilpotent adjacency matrix over ${\mathcal{C}\ell_n}^{\mathrm{nil}} \otimes {\mathcal{C}\ell_{|E|}}^{\mathrm{nil}}$. Then, for any fixed $m \geq 1$, determining the decomposition of G into m edge-disjoint cycles requires $O(n^{\beta+1} \log n)$ Clops.*

Proof. As in the proof of Corollary 2.12, computing $\sum_{k=3}^{n} s_k \frac{\tau_k}{2k}$ has $\mathcal{C}\ell$ complexity $O(n^{\beta+1} \log n)$. Then computing $\left(\vartheta \left(\sum_{k=3}^{n} s_k \frac{\tau_k}{2k} \right) \right)^m$ has $\mathcal{C}\ell$ complexity $O(n^{\beta+1} \log n + \log m) = O(n^{\beta+1} \log n)$. □

Example 2.23. The cycle decompositions of the complete graph K_5 are recovered.



The nilpotent adjacency matrix is

$$\Lambda = \begin{pmatrix} 0 & \zeta_2\gamma_1 & \zeta_3\gamma_2 & \zeta_4\gamma_3 & \zeta_5\gamma_4 \\ \zeta_1\gamma_1 & 0 & \zeta_3\gamma_5 & \zeta_4\gamma_6 & \zeta_5\gamma_7 \\ \zeta_1\gamma_2 & \zeta_2\gamma_5 & 0 & \zeta_4\gamma_8 & \zeta_5\gamma_9 \\ \zeta_1\gamma_3 & \zeta_2\gamma_6 & \zeta_3\gamma_8 & 0 & \zeta_5\gamma_{10} \\ \zeta_1\gamma_4 & \zeta_2\gamma_7 & \zeta_3\gamma_9 & \zeta_4\gamma_{10} & 0 \end{pmatrix}. \tag{2.23}$$

Let D_t denote the operator of differentiation with respect to t, and denote the m^{th} order differentiation operator by $D_t^{(m)}$. Terms of the power series expansion of $\left\langle \exp\left(t\vartheta \left(\sum_{k=3}^{5} s_k \frac{\tau_k}{2k} \right) \right) \right\rangle_{10}$ can be extracted with Mathematica to recover powers

of $\vartheta\left(\sum_{k=3}^{5} s_k \frac{\tau_k}{2k}\right)$. Mathematica calculations yield the following:

$$\left\langle\left\langle D_t^{(1)}\left\langle \exp\left(t\vartheta\left(\sum_{k=3}^{5} s_k\frac{\tau_k}{2k}\right)\right)\right\rangle_{10}\Big|_{t=0}\right\rangle\right\rangle = 0 \tag{2.24}$$

$$\left\langle\left\langle D_t^{(2)}\left\langle \exp\left(t\vartheta\left(\sum_{k=3}^{5} s_k\frac{\tau_k}{2k}\right)\right)\right\rangle_{10}\Big|_{t=0}\right\rangle\right\rangle = 12{s_5}^2 \tag{2.25}$$

$$\left\langle\left\langle D_t^{(3)}\left\langle \exp\left(t\vartheta\left(\sum_{k=3}^{5} s_k\frac{\tau_k}{2k}\right)\right)\right\rangle_{10}\Big|_{t=0}\right\rangle\right\rangle = 90{s_3}^2 s_4 \tag{2.26}$$

$$\left\langle\left\langle D_t^{(4)}\left\langle \exp\left(t\vartheta\left(\sum_{k=3}^{5} s_k\frac{\tau_k}{2k}\right)\right)\right\rangle_{10}\Big|_{t=0}\right\rangle\right\rangle = 0. \tag{2.27}$$

The results are now interpreted:

K_5 itself is not a cycle, so the first expression yields zero.

The only decompositions of K_5 into pairs of cycles are $6 = 12/(2!)$ decompositions into pairs of five-cycles. That is, $K_5 = C_5 \cup C_5$ with multiplicity six.

The only decompositions into triples of cycles are $15 = 90/(3!)$ decompositions of the form $K_5 = C_3 \cup C_3 \cup C_4$, i.e., a pair of three-cycles and a four-cycle.

There are no decompositions of K_5 into four or more disjoint cycles.

Similar to a cycle decomposition of a graph, which is a partitioning of the graph's edges, a cycle cover of a graph is a partitioning of the graph's vertices such that each vertex is contained in exactly one cycle.

Definition 2.24. A *cycle cover* of a graph G is defined as a pairwise disjoint collection of cycles $\{C_j\}$ such that each vertex of G is contained in exactly one of the cycles.

Theorem 2.25. *Let Λ be the nilpotent adjacency matrix of a finite graph G on n vertices. Then, letting C denote the number of cycle covers of G,*

$$\left\langle\left(\sum_{\ell=1}^{\lfloor n/3\rfloor}\frac{1}{\ell!}\vartheta\left(\sum_{m=3}^{n}\frac{\tau_m}{2\,m}\right)^{\ell}\right), \gamma_{[|E|]}\right\rangle = C. \tag{2.28}$$

Proof. For each $3 \le m \le n$, $\frac{\tau_m}{2\,m}$ denotes the algebraic sum of multivectors representing m-cycles in G. Computing the ℓ^{th} power of the sum of these reveals pairwise disjoint ℓ-tuples of cycles of all lengths, each appearing with multiplicity $\ell!$ Because G has n edges, the maximum number of ℓ-tuples of disjoint cycles in G is $\lfloor n/3\rfloor$. Summing over all admissible values of ℓ and considering the coefficient of $\gamma_{[|E|]}$ to ensure that all edges of G are covered, the number of cycle covers of G is recovered. □

For each $m = 3, \ldots, n$, computing τ^m requires $O(n^\beta \log m)$ Clops. Hence, computing $\sum_{m=3}^{n} \frac{\tau_m}{2\,m}$ requires $O(n^{\beta+1} \log n)$ Clops. For each $\ell = 1, \ldots, \lfloor n/3 \rfloor$, computing $\left(\sum_{m=3}^{n} \frac{\tau_m}{2\,m} \right)^\ell$ then requires $O(n^{\beta+1} \log n \log \ell)$ Clops. Finally, summing over ℓ, $\sum_{\ell=1}^{\lfloor n/3 \rfloor} \frac{1}{\ell!} \left(\sum_{m=3}^{n} \frac{\tau_m}{2\,m} \right)^\ell$ requires $O(n^{\beta+2} \log n \log n)$ Clops. It therefore follows from the results obtained thus far that counting the disjoint cycle covers of a graph on n vertices requires $O(n^{\beta+2} \log n \log n)$ Clops. In the next section, this result is improved by another method.

3. Other Problems

In addition to graph problems, properties of ${\mathcal{C}\ell_n}^{\mathrm{nil}}$ and ${\mathcal{C}\ell_n}^{\mathrm{idem}}$ make them useful for other types of combinatorial problems.

3.1. Computing the Permanent

The problem of computing the permanent of a matrix is known to be ♯P-complete [25], [26]. Methods of approximating the permanent using Clifford algebras have also been discussed [27].

The current authors' methods allow one to directly compute the permanent of a matrix with $\mathcal{C}\ell$ computational complexity $O(n)$.

Let $M = (m_{ij})_{n \times n}$ be an arbitrary matrix. Let $\{\gamma_i\}$ and $\{\zeta_j\}$, $1 \leq i, j \leq n$ denote commutative null-square generators of ${\mathcal{C}\ell_n}^{\mathrm{nil}}$ and define

$$a = \sum_{i,j=1}^{n} m_{ij}\, \gamma_i\, \zeta_j \in {\mathcal{C}\ell_n}^{\mathrm{nil}} \otimes {\mathcal{C}\ell_n}^{\mathrm{nil}}. \tag{3.1}$$

Proposition 3.1. *Let M, n, and a be defined as above. Then,*

$$\left\langle a^n, \gamma_{[n]}\, \zeta_{[n]} \right\rangle = n!\, \mathrm{Perm}(M). \tag{3.2}$$

Proof. Let M be an $n \times n$ matrix and consider the following definition of the permanent:

$$\mathrm{Perm}(M) = \sum_{\sigma \in \mathrm{S}_n} \prod_{i=1}^{n} m_{i\,\sigma(i)}, \tag{3.3}$$

where S_n is the symmetric group of order $n!$

Now,

$$a^n = \left(\sum_{i,j=1}^{n} m_{ij}\,\gamma_i\,\zeta_j\right)^n = \left(\sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}\,\gamma_i\,\zeta_j\right)^n$$

$$= \sum_{k_1+\cdots+k_n=n} \binom{n}{k_1,\ldots,k_n} \prod_{i=1}^{n} \left(\gamma_i \sum_{j=1}^{n} m_{ij}\,\zeta_j\right)^{k_i}$$

$$= \binom{n}{1,1,\ldots,1} \prod_{i=1}^{n} \left(\sum_{j=1}^{n} m_{ij}\,\gamma_i\,\zeta_j\right)$$

$$= n! \sum_{\pi\in S_n} \prod_{i=1}^{n} m_{i\,\pi(i)}\,\gamma_i\,\zeta_{\pi(i)} = n!\,\mathrm{Perm}(M)\,\gamma_{[n]}\,\zeta_{[n]}. \quad (3.4)$$

□

Corollary 3.2. *Computing the permanent of an $n \times n$ matrix is of $\mathcal{C}\ell$ complexity $O(n)$.*

Given a graph $G = (V, E)$ on n vertices, a *cycle cover* of G is a collection of cycles $\{C_1, \ldots, C_k\}$ contained as subgraphs of G such that each vertex of G is contained in exactly one of the cycles.

Recall the definition (3.3) of the permanent of an $n \times n$ matrix. Each cyclic permutation in S_n corresponds to a unique cycle in the graph via the adjacency matrix. Since every $\sigma \in S_n$ can be written uniquely as a disjoint product of cycles, it follows immediately that the permanent of the adjacency matrix counts the number of cycle covers of the graph.

Corollary 3.3 (Complexity of cycle covers). *Counting the cycle covers of a finite graph on n vertices is of $\mathcal{C}\ell$ complexity $O(n)$.*

Given a graph $G = (V, E)$, a *matching* of G is a subset $E_1 \subset E$ of the edges of G having the property that no pair of edges in E_1 shares a common vertex. The largest possible matching on a graph with n vertices consists of $n/2$ edges, and such a matching is called a *perfect matching*.

A graph $G = (V, E)$ is said to be *bipartite* if its vertex set V can be partitioned into a pair of disjoint sets V_1, V_2 such that $u, w \in V_\ell \Rightarrow \{u, w\} \notin E$ for $\ell = 1, 2$ (cf. Figure 1).

It follows from definitions that only graphs with even numbers of vertices have perfect matchings and that bipartite graphs contain only cycles of even length.

It is well-known and not difficult to see that the permanent of the adjacency matrix of a bipartite graph counts the perfect matchings of the graph. Given a cycle of even length, a perfect matching for the cycle is obtained by choosing alternating edges.

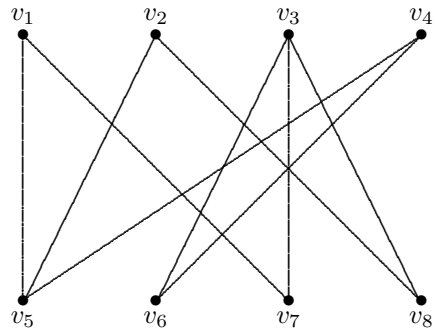


FIGURE 1. A bipartite graph on 8 vertices.

Corollary 3.4. *Counting the perfect matchings of a bipartite graph is of $\mathcal{C}\ell$ complexity $O(n)$.*

3.2. The Set Packing and Set Covering Problems

The following two problems are among the original 21 NP-complete problems of Karp [28]. They are moved to class P in the $\mathcal{C}\ell$ context.

Theorem 3.5 (Set covering problem). *Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be a collection of subsets of the n-set $\{1, 2, \ldots, n\}$. In the $\mathcal{C}\ell$ context, the problem of determining the minimum value of k for which there exists a collection $\{S_{j_1}, \ldots, S_{j_k}\} \subseteq \mathcal{S}$ satisfying $\bigcup_{\ell=1}^{k} S_{j_\ell} = \{1, 2, \ldots, n\}$ has $\mathcal{C}\ell$ complexity $O(m \log k)$.*

Proof. Let $\alpha = \sum_{j=1}^{m} \varepsilon_{S_j} \in {\mathcal{C}\ell_n}^{\mathrm{idem}}$. That is, α is a sum of basis blades of ${\mathcal{C}\ell_n}^{\mathrm{idem}}$ indexed by the sets S_j. Then there exists a collection $\{S_{j_1}, \ldots, S_{j_k}\} \subseteq \mathcal{S}$ such that $\bigcup_{\ell=1}^{k} S_{j_\ell} = \{1, 2, \ldots, n\}$ if and only if $\langle \alpha^k, \varepsilon_{[n]} \rangle \neq 0$. Checking each $k = 1, 2, \ldots, m$ requires at most m iterations. □

Theorem 3.6 (Set packing problem). *Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be a collection of subsets of the n set $\{1, 2, \ldots, n\}$. In the $\mathcal{C}\ell$ context, the problem of determining whether there exists a pairwise disjoint collection $\{S_{j_1}, \ldots, S_{j_k}\} \subseteq \mathcal{S}$ such that $\bigcup_{\ell=1}^{k} S_{j_\ell} = \{1, 2, \ldots, n\}$ has $\mathcal{C}\ell$ complexity $O(\log k)$.*

Proof. Let $\alpha = \sum_{j=1}^{m} \zeta_{S_j} \in {\mathcal{C}\ell_n}^{\text{nil}}$. In other words, α is a sum of basis blades of ${\mathcal{C}\ell_n}^{\text{nil}}$ indexed by the sets S_j. Then there exists a pairwise disjoint collection $\{S_{j_1}, \ldots, S_{j_k}\} \subseteq \mathcal{S}$ if and only if $\alpha^k \neq 0$. □

4. Conclusion

Given the existence of a computer architecture capable of dealing naturally with geometric objects, a natural measure of algorithmic complexity is the number of Clifford operations ($\mathcal{C}\ell$ops) required by the algorithm. A number of combinatorial problems of complexity class NP are of polynomial complexity in terms of $\mathcal{C}\ell$ops required. We assert that a Clifford computer would have natural advantages for solving an assortment of combinatorial and graph-theoretic problems.

In addition, as Aerts and Czachor have shown [12], a Clifford computer would be able to implement quantum computing algorithms while avoiding the noise and error-correcting concerns associated with quantum systems.

References

[1] J. Lasenby, W. J. Fitzgerald, C. J. L. Doran, A.N. Lasenby, *New geometric methods for computer vision*. Int. J. Comp. Vision, **36** (1998), 191-213.

[2] H. Li, *Clifford algebra approaches to automated geometry theorem proving*. Mathematics Mechanization and Applications, X.-S. Gao and D. Wang (eds.), Academic Press, London, 2000, pp. 205-230.

[3] C. Perwass, C. Gebken, G. Sommer, *Estimation of geometric entities and operators from uncertain data*. Lecture Notes in Computer Science **3363**, Springer, Berlin, 2005.

[4] R. Schott, G.S. Staples, *Nilpotent adjacency matrices and random graphs*. Ars Combinatoria, To appear.

[5] R. Schott, G.S. Staples, *Nilpotent adjacency matrices, random graphs, and quantum random variables*. J. Phys. A: Math. Theor., **41** (2008), 155205.

[6] G.S. Staples, *Clifford-algebraic random walks on the hypercube*. Advances in Applied Clifford Algebras, **15** (2) (2005), 213-232.

[7] G.S. Staples, *Norms and generating functions in Clifford algebras*. Advances in Applied Clifford Algebras, **18** (1) (2008), 75-92.

[8] G.S. Staples, *Graph-theoretic approach to stochastic integrals with Clifford algebras*. J. Theor. Prob., **20** (2007), 257-274.

[9] D. Applebaum, *Fermion stochastic calculus in Dirac-Fock space*. J. Phys. A: Math. Theor., **28** (1995), 257-270.

[10] D. Applebaum, R. Hudson, *Fermion Itô's formula and stochastic evolutions*. Commun. Math. Phys., **96** (1984), 473-96.

[11] C. Barnett, R. Streater, I. Wilde, *The Itô-Clifford integral I*. J. Functional Analysis. **48** (1982), 172-212.

[12] D. Aerts, M. Czachor, *Cartoon computation: quantum-like computing without quantum mechanics*. J. Phys. A: Math. Theor. **40** (2007), F259-F263.

[13] C. Perwass, *The CLU Project web page*. `http://www.perwass.de/cbup/clu.html`

[14] P. Leopardi, *The GluCat Home Page*. `http://glucat.sourceforge.net/`

[15] D. Fontijne, T. Bouma, L. Dorst, *GAIGEN: a geometric algebra implementation generator*. University of Amsterdam, NL, July 2002. `http://www.science.uva.nl/ga/gaigen`

[16] R. Abłamowicz, B. Fauser, *CLIFFORD - A Maple Package for Clifford Algebra Computations*. `http://math.tntech.edu/rafal/`

[17] C. Perwass, C. Gebken, G. Sommer, *Implementation of a Clifford algebra coprocessor design on a field programmable gate array*. Clifford Algebras Applications to Mathematics, Physics, and Engineering, Progress in Mathematical Physics **34**, Birkhäuser, Boston, 2004.

[18] B. Mishra, P. Wilson, *Color edge detection hardware based on geometric algebra*. `http://eprints.ecs.soton.ac.uk/13188/`

[19] B. Mishra, P. Wilson, *Hardware implementation of a geometric algebra processor core*. Proceedings of ACA 2005, IMACS, Int. Conf. on Advancement of Computer Algebra, Nara, Japan, 2005. `http://eprints.ecs.soton.ac.uk/10957/`

[20] S. Franchini, A. Gentile, M. Grimaudo, C.A. Hung, S. Impastato, F. Sorbello, G. Vassallo, S. Vitabile, *A sliced coprocessor for native Clifford algebra operations*. Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007), (2007), 436-439.

[21] A. Gentile, S. Segreto, F. Sorbello, G. Vassallo, S. Vitabile, V. Vuollo, *CliffoSor, an innovative FPGA-based architecture for geometric algebra*. Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA 2005), 211-217.

[22] A. Gentile, S. Segreto, F. Sorbello, G. Vassallo, S. Vitabile, V. Vuollo, *CliffoSor: a parallel embedded architecture for geometric algebra and computer graphics*. Proceedings of the IEEE International Workshop on Computer Architecture for Machine Perception (CAMP 2005), 90-95, IEEE Computer Society Press.

[23] I. Porteous, *Lecture 2: Mathematical structure of Clifford algebras*. Lectures on Clifford (Geometric) Algebras and Applications, R. Abłamowicz, G. Sobczyk, Eds., Birkhäuser, Boston, 2003.

[24] D. Coppersmith, S. Winograd, *Matrix multiplication via arithmetic progressions*. Journal of Symbolic Computation **9** (1990), 251280.

[25] A. Ben-Dor, S. Halevi, *Zero-one permanent is ♯P-complete, a simpler proof*. Proceedings of the 2nd Israel Symposium on the Theory and Computing Systems, (1993), 108-117.

[26] L. Valiant, *The complexity of computing the permanent*. Theoretical Computer Science **8** (1979), 189-201.

[27] S. Chien, L. Rasmussen, A. Sinclair, *Clifford algebras and approximating the permanent*. Journal of Computer and System Sciences **67** (2003), 263-290.

[28] R. M. Karp, *Reducibility among combinatorial problems*. Complexity of Computer Computations, Plenum, New York 1972, 85-103.

Acknowledgment

The authors are grateful to the anonymous referee for a number of helpful suggestions. The second author thanks Philip Feinsilver for useful discussion about the matrix permanent.

René Schott
IECN and LORIA Université Henri Poincaré-Nancy I
BP 239
54506 Vandoeuvre-lès-Nancy
France
e-mail: schott@loria.fr

G. Stacey Staples
Department of Mathematics and Statistics
Southern Illinois University at Edwardsville
Edwardsville, IL 62026-1653
USA
e-mail: sstaple@siue.edu

Received: July 4, 2008.
Accepted: October 1, 2008.