SUMMER PROJECT

JAIUNG JUN AND MATTHEW PISANO

Abstract.

1. Introduction

We will focus on Research Project 11 in [GK20]. The following are more specific directions that we plan to pursue.

Goal (8/1/2022)**.**

(1) Prove/disprove: for an oriented graph G, one always has $\mathrm{Pic}(G) = \mathbb{Z} \times \mathrm{Jac}(G)$, i.e., as a finitely generated abelian group, the rank of $\mathrm{Pic}(G)$ is 1.
(2) Prove/disprove: for C_n, and $0 \leq m \leq n$, one can always find an orientation of C_n so that $\mathrm{Jac}(C_n) = \mathbb{Z}_m$ (with the orientation).
(3) Prove/disprove: for an oriented graph G, if $v_0 \in V(G)$ is a sink (or a source) and G' is the graph obtained by reserving the direction for all arrows adjacent to v_0 from G, then $\mathrm{Jac}(G) = \mathrm{Jac}(G')$. (Note: we believe that this should be true for at least some classes of graphs such as cyclic graphs.)
(4) Prove/disprove: for an oriented planar graph G and its planar dual (should be defined) $\hat{G}$, one has $\mathrm{Jac}(G) = \mathrm{Jac}(\hat{G})$.
(5) Prove/disprove: for oriented graphs G_1, G_2, let G be the graph obtained by gluing G_1 and G_2 along one vertex. Then $\mathrm{Jac}(G) = \mathrm{Jac}(G_1) \times \mathrm{Jac}(G_2)$.

2. Preliminaries

2.1. Chip Firing. The game at the heart of this paper is the Chip-Firing game. When a game is started, each vertex on a graph is assigned a certain number of chips. During play, chips can be lent or borrowed at each node where one or more chips are either sent or received along each outgoing edge equally. In the case of a directed graph, vertices can only interact with another along an outgoing or bidirectional edge. The game is won once every vertex has a positive number of chips (i.e., this vertex is not in debt).

2.2. Divisors and Equivalence Relations. In the study of this game a **Divisor** of a graph ($\mathrm{Div}(G)$) is an integer vector $v \in \mathbb{Z}^n$ where n is the number of vertices in the graph. The i^{th} of element of the vector v is the number of chips on the i^{th} vertex of the graph. Two divisors have an **Equivalence Relation** ($\sim$) if one divisor can be gotten from the other by a finite series of lending or borrowing moves $D_1 \sim D_2 \leftrightarrow (D_1 \xleftrightarrow{\text{moves}} D_2)$. An **Equivalence Class** *[D]* is the set of all divisors that are equivalent to each other, $[D] = \{D_i \mid D_i \sim D\}$.

2020 *Mathematics Subject Classification.* 05C50, 05C76.

Key words and phrases. Jacobian of a graph, sandpile group, critical group, chip-firing game, gluing graphs, cycle graph, Tutte polynomial, Tutte's rotor construction.

2.3. The Picard Group and The Jacobian. The **Picard Group** of a graph $\text{Pic}(G)$ is the set of all equivalence classes that the divisors of that graph can be a part of. The **Jacobian** of a graph $\text{Jac}(G)$ is a subset of $\text{Pic}(G)$ such that every divisor in each equivalency class has a degree of 0 where the degree of a divisor $\deg(D)$ is the sum of each of the divisor's elements. If a divisor is in one of the Jacobian's classes, it can be made winning after a finite series of moves.

3. Propositions

3.1. Calculating Sinks and Sources. Vertices on a directed graph can be classified in three ways, as a sink, source, or neither depending on the direction of the edges connected to it. A sink is a vertex where all edges are directed into that vertex, a source is a vertex where all edges are directed away from that vertex and a vertex is neither when it has a mixture of the two. The number of sinks and sources can be calculated inductively by reconstructing the original graph from a single vertex.

Beginning with any two vertices that are adjacent on the original graph, observe their connecting edge. If the edge is directed, the count of sinks and sources begins at 1 for both. If the edge is undirected (bidirectional), the count begins at zero. Next, pick a vertex adjacent to one of the original edges and observe its connection. From this point on, when a new edge is observed the number of sinks and sources changes on a series of rules. For these rules, let the vertex being added to be V_1 and the vertex being added be V_2.

(1) If V_1 is neither a sink nor a source.
   (a) Adding an edge directed towards V_2 adds a sink.
   (b) Adding an edge directed away from V_2 adds a source.
   (c) Adding a bidirectional edge between V_1 and V_2 adds nothing.
(2) If V_1 is a source.
   (a) Adding an edge directed towards V_2 adds a sink.
   (b) Adding an edge directed away from V_2 adds nothing (Here the number of sources stays the same as V_1 is no longer a source but V_2 now is).
   (c) Adding a bidirectional edge between V_1 and V_2 removes a source.
(3) If V_1 is a sink.
   (a) Adding an edge directed towards V_2 adds nothing (See 2b).
   (b) Adding an edge directed away from V_2 adds a source.
   (c) Adding a bidirectional edge between V_1 and V_2 removes a sink.

When accounting for a vertex that has multiple connections, apply these rules for every V_{1i} that V_2 is connected to, taking into account all the edges of V_2 when evaluating its class. After every vertex has been accounted for, the number of sinks and sources will have been calculated in polynomial time between $O(n)$ at the average case and $O(n^2)$ for the case of a connected graph.

3.2. Using Sinks and Sources to Calculate $Rank(\text{Pic}(G))$**.** $\text{Pic}(G)$ is often in the form $\mathbb{Z}_1 \times \cdots \times \mathbb{Z}_n \times \mathbb{Z}^m$ where m is the rank of the Picard group. When calculating $\text{Pic}(G)$, its $Rank()$ always the number of sinks for any arbitrary graph. $Rank(\text{Pic}(G))$ is represented in the smith normal form of the laplacian as the number of all zero rows. For all $Rank(\text{Pic}(G)) \geq 2$, the number of all zero rows in the smith normal form matches the number of all zero rows in the laplacian matrix of G or the number of sink vertices in G. The rank of a graph's Picard group can be found by applying 3.1 to keep track of the sinks while recreating the original graph.

3.3. Directed Cyclic Jacobians. Proposition: For C_n, and $0 \leq m \leq n$, one can always find an orientation of C_n so that $\text{Jac}(C_n) = \mathbb{Z}_m$ (with the orientation). Proof: the proof for this proposition was proven incompletely up to C_{17} using a brute force method of checking all oriented configurations of directed edges until all $\{\text{Jac}(C_n) = \mathbb{Z}_m \mid 0 \leq m \leq n\}$ have been found. The first C_3 C_{17} were proven in $O(3^n)$ time. The algorithm used did not have to check all permutations, however, only a very cmall portion. This resulted in each portion being $\frac{1}{3}$ of the portion of the previous graph. Due to this, calculations for the first 17 graph was calculated in closer to $O(x^n)$ polynomial time.

References

[GK20] Darren Glass and Nathan Kaplan. Chip-firing games and critical groups. In *A Project-Based Guide to Undergraduate Research in Mathematics*, pages 107–152. Springer, 2020.

DEPARTMENT OF MATHEMATICS, STATE UNIVERSITY OF NEW YORK AT NEW PALTZ, NY 12561, USA
Email address: `junj@newpaltz.edu`

STATE UNIVERSITY OF NEW YORK AT BINGHAMTON, NY 13902, USA
Email address: `pisanom1@newpaltz.edu`