A calculus for monomials in Chow group of zero cycles in the moduli space of stable curves

Jiayue Qi

DK-Report No. 2020-11 09 2020

Supported by

Austrian Science Fund (FWF) Upper Austria

FWF Der Wissenschaftsfonds.

LAND OBERÖSTERREICH

DK sponsors:

- **Johannes Kepler University Linz** (JKU)
- **Austrian Science Fund** (FWF)
- **Upper Austria**

A calculus for monomials in Chow group of zero cycles in the moduli space of stable curves

Jiayue Qi*
Department of Mathematics
University of Linz
4040 Linz, Austria
jiayue.qi@dk-compmath.jku.at

Abstract

We introduce an algorithm for computing the value of all monomials in the Chow group of zero cycles in the moduli space of stable curves.

Keywords: monomials in Chow ring, monomial value, tree representations, recursive algorithm on forest

Chow rings and Chow groups are essential tools in intersection theory. Multiplication of elements in the Chow ring corresponds to the intersection of cycles (formal sums of subvarieties). For a projective variety X of dimension r, the Chow ring $A^*(X)$ is the direct sum of $r+1$ groups, each composed of cycles of a given dimension. In particular, the Chow group $A^r(X)$ is the group of cycles of dimension zero. The problem that we study in this paper shows up as a sub-problem for counting the realization of Laman graphs (minimally rigid graphs) on a sphere [1]. However, our algorithm applies in a wider context – in computing the value of any monomials in this Chow group.

Let $n \in \mathbb{N}$, $n \geq 3$, set $N := \{1, \ldots, n\}$. $\mathcal{M}_n$ denotes the moduli space of stable n-pointed curves of genus zero. It is a well-studied object in algebraic geometry. A bipartition $\{I, J\}$ of N where both cardinality of I and J are at least 2 is called a **cut**; I and J are **parts** of this cut. For every cut $\{I, J\}$, there is a variety $D_{I,J}$ in $\mathcal{M}_n$; denote by $\delta_{I,J}$ its corresponding element in the Chow ring. Note that $D_{I,J} = D_{J,I}$ and $\delta_{I,J} = \delta_{J,I}$. It is a graded ring – denote it as $A^*(n)$ – we have $A^*(n) = \bigoplus_{r=0}^{n-3} A^r(n)$. These homogeneous components are defined as Chow groups (of M_n); $A^r(n)$ is the **Chow group of rank** r. It is known that $A^r(n) = \{0\}$ for $r > n-3$ and $A^{n-3}(n) \cong \mathbb{Z}$. We denote this isomorphism as $\int : A^{n-3}(n) \to \mathbb{Z}$.

The set $\{\delta_{I,J} \mid \{I, J\} \text{ is a cut}\}$ generates group $A^1(n)$, and also the whole ring $A^*(n)$ (when they are used as ring generators). Then, $\prod_{i=1}^{n-3} \delta_{I_i,J_i}$ can be viewed as an element in $A^{n-3}(n)$, since we are in a graded ring. Denote $M := \prod_{i=1}^{n-3} \delta_{I_i,J_i}$, we define the **value of** M to be $\int(\prod_{i=1}^{n-3} \delta_{I_i,J_i})$.

In this paper we calculate the value of a given monomial $M = \prod_{i=1}^{n-3} \delta_{I_i,J_i}$.

There are linear and quadratic relations between the generators. Denote $\epsilon_{ij|kl} := \sum_{i,j\in I,k,l\in J} \delta_{I,J}$. Then we have the equalities $\epsilon_{ij|kl} = \epsilon_{il|kj} = \epsilon_{ik|jl}$, we call it **Keel's linear relation** [2]. Among the

*The research was funded by the Austrian Science Fund (FWF): W1214-N15, project DK9.

generators of $A^*(n)$, we say that the two generators $\delta_{I_1,J_1}, \delta_{I_2,J_2}$ fulfill **Keel's quadratic relation** [2] if the following four conditions hold: $I_1 \cap I_2 \neq \emptyset$; $I_1 \cap J_2 \neq \emptyset$; $J_1 \cap I_2 \neq \emptyset$; $J_1 \cap J_2 \neq \emptyset$. In this case, $\delta_{I_1,J_1} \cdot \delta_{I_2,J_2} = 0$.

Then we know that an easy case for our value computing problem is when two factors of the monomial fulfill Keel's quadratic relation – we simply get zero. Hence we only need to consider the monomials where no two factors fulfill Keel's quadratic relation; we call this type of monomial **tree monomial** since there is a one-to-one correspondence between these monomials and *loaded trees* (see Theorem 0.2).

Definition 0.1. A **loaded tree with n labels and k edges** is a tree (V, E) together with a labeling function $h : V \to 2^N$ and an edge multiplicity function $m : E \to \mathbb{N}^+$ such that the following three conditions hold:

1. $\{h(v)\}_{v \in V, h(v) \neq \emptyset}$ form a partition of N;
2. $\sum_{e \in E} m(e) = k$;
3. For every $v \in V$, $\deg(v) + |h(v)| \geq 3$, note that here multiple edges are only counted once for the degree of its incident vertices.

We define the **monomial of a given loaded tree** as follows: For each edge we collect the labels on one side of it to form I and labels on the other side to form J. Then we say that $\{I, J\}$ is the corresponding cut for this edge. Monomial of this given loaded tree is $\prod_{i=1}^{k} \delta_{I_i,J_i}$; each edge of the tree contributes to the monomial a factor $\delta_{I,J}$ if $\{I, J\}$ is the corresponding cut for this edge. It is well-defined and each loaded tree has a unique monomial representation.

Theorem 0.2. *There is a one-to-one correspondence between tree monomials $M = \prod_{i=1}^{k} \delta_{I_i,J_i}$ and loaded trees with n labels and k edges, where $I_i \cup J_i = N$ for all $1 \leq i \leq k$.*

It is trivial to transfer to a monomial from a given loaded tree. Correspondingly to the above theorem, we provide an algorithm transferring from a tree monomial to a loaded tree. See Algorithm 1. Note that in this algorithm we restrict the ambient group to be $A^{n-3}(n)$, but it can apply to a much wider context – transferring any tree monomial to its corresponding loaded tree. After this step, we start from that loaded tree, go through several transformations, until finally value of the given tree monomial is obtained.

Because of this one-to-one correspondence, we define **value of a loaded tree** T as $\int(M_T)$, where M_T is its corresponding monomial of T; denote it as $\int(T)$. Given a loaded tree with n labels and $n-3$ edges, our calculus calculates its value. The calculus contains several steps. First: Check whether the given monomial contains a pair which fulfills Keel's quadratic relation. If yes, the value of the monomial is 0; if no, we continue with the second step. We need the following known result to explain the second step.

Theorem 0.3. *If all factors are distinct in $M = \Pi_{i=1}^{n-3} \delta_{I_i,J_i}$, then $\int(M) = 1$. We call this type of tree monomial **clever monomial** and its corresponding loaded tree **clever tree**.*

In the second step, we check whether the given monomial is a clever monomial. If yes, we know that the value is 1; if no, we go to the third step. The third step as well contains several sub-steps. Main idea behind is that by using Keel's linear relation, we can substitute some higher-power factors. And hopefully finally get a sum of clever monomials (maybe with a negative sign). Then the number of of clever monomials should be the absolute value of the given monomial. Based on this idea, we have an algorithm for calculating the value of all tree monomials in $A^{n-3}(n)$. Here we

Algorithm 1: monomial to tree

```
input  : a tree monomial $M$ in $A^{n-3}(n)$
output: a loaded tree with $n$ labels and $n-3$ edges
$C \leftarrow$ collection of any cut that corresponds to some factor of $M$;
$P \leftarrow$ collection of all the parts of cuts in $C$;
$c \leftarrow$ any element $c = \{I, J\} \in C$;
for each element $p \in P \setminus \{I, J\}$ do
    if $p \subset I$ or $p \subset J$ then
        $c := c \cup \{p\}$
    end if
end for
$H \leftarrow$ the Hasse diagram of elements in $c$ with respect to set containment order;
Consider $H$ as a graph $(V, E)$;
for each vertex $V$ of $H$ do
    Define labelling set $h(V)$ as its corresponding element in $c$;
    Update the labelling set: $h(V) := h(V) \setminus h(V_1)$ if $V_1$ is less than $V$ in $H$
end for
$E := E \cup \{\{I, J\}\}$;
Attach this labelling function $h$ to $H$;
Set the multiplicity function value $m(e)$ for each edge $e$ as the power of its corresponding
  factor in $M$;
return $H = (V, E, h, m)$
```

give the sketch of the second step: **Input:** a loaded tree with n labels and $n-3$ edges. **Output:** a natural number. (1) Transfer the loaded tree to a **semi-redundancy tree**. (2) Calculate the **sign of the tree value**. (3) Construct a **redundancy forest** from the semi-redundancy tree. (4) Apply a recursive algorithm to this redundancy forest, obtaining the absolute tree value. (5) Product of the sign and absolute value gives us tree value.

Now we explain those terminologies. Given a loaded tree $LT = (V, E, h, m)$. Define a weight function $w : V \cup E \to \mathbb{N}$ as follows: For any $v \in V$, $w(v) := \deg(v) + |h(v)| - 3$. From Definition 0.1 we see that the weight of any vertex must be non-negative. For any $e \in E$, $w(e) := m(e) - 1$. From Definition 0.1 we see that the weight of any edge is also non-negative. Then the **semi-redundancy tree** (of LT) is $SRT := (V, E, w)$. Start from this semi-redundancy tree, let S be the sum of vertex weight (or edge weight) of LT. **Sign of the tree value** of loaded tree LT is $(-1)^S$. It is not hard to verify that weight sum of edges and of vertices are the same when the given loaded tree has number of labels three more than the sum of multiplicity of its edges.

Next, how do we transfer a semi-redundancy tree (V, E, w) (assume $LT = (V, E, h, m)$) to a redundancy forest? Replace each edge by a length-two edge with a new vertex connecting them with the same weight as the replaced edge. Then we obtain the redundancy tree (of loaded tree LT) $RT := (V \cup E, E_1, w_1)$. Union of subtrees of RT such that no vertex has weight zero is the **redundancy forest** of LT.

Starting from the redundancy forest, we can compute the absolute value using a recursive algorithm. Let $RF = (V, E, w)$ be the redundancy forest of a loaded tree LT. We define the **value of** RF as follows: Pick any leaf $l \in V$ of this forest, denote the unique parent of l as l_1. If $w(l) > w(l_1)$, return 0 and terminate the process; otherwise, remove l from RF and assign weight $(w(l_1) - w(l))$ to l_1, replacing its previous weight. Denote this new forest as RF_1. Value of RF is

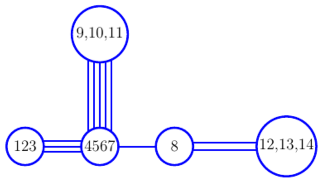

Figure 1: This is a loaded tree LT with 14 labels and 11 edges. Labels are tagged in black.

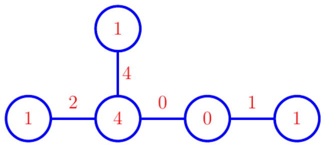

Figure 2: This shows the semi-redundancy tree SRT of the loaded tree LT in Figure 1. Weight function is marked in red.

the product of binomial coefficient $\binom{w(l_1)}{w(l)}$ and the value of RF_1. Whenever we reach a degree-zero vertex, if it has non-zero weight, return 0 and terminate the process; otherwise return 1. Product of absolute value of the corresponding redundancy forest of LT and sign of its tree value gives us the value of LT. In the sequel, we explain this process with an example, to show it more intuitively.

Example 0.4. Given a loaded tree LT, see Figure 1. Follow the definition (construction) of semi-redundancy tree, we obtain Figure 2. Sum of vertex weight $S = 1+4+1+0+1 = 7$, so the sign of LT value is $(-1)^7 = -1$. From SRT, follow the construction for the redundancy forest, we obtain Figure 3. Finally we get the absolute value of RF as $[\binom{1}{1} \times 1] \times [\binom{2}{1} \times \binom{4}{1} \times \binom{4}{3} \times \binom{1}{1} \times 1] = 32$. Combining with the sign -1, we obtain the value of LT as -32.

It is not hard to verify that the above process terminates. It is well-defined – the result is independent of the sequence of leaves we choose (to apply the recursive algorithm on the redundancy forest) – because of the following identity in binomial coeffients:

$$\binom{c}{a} \cdot \binom{c-a}{b} \equiv \binom{c}{b} \cdot \binom{c-b}{a}.$$

Therefore it is indeed an algorithm. We call it **forest algorithm**.

Theorem 0.5. *The forest algorithm is correct, i.e., it indeed calculates the value of a given loaded tree, or equivalently, the value of its corresponding (tree) monomial.*

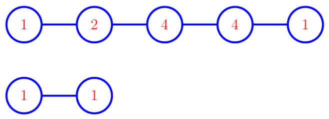

Figure 3: This is the redundancy forest RF of loaded tree LT in Figure 1. Weight function is marked in red.

Acknowledgement

I thank Josef Schicho for introducing to me the problem background and posting to me the spherical-Laman-graph counting problem. I thank Josef Schicho for general discussion and the proof of correctness of forest algorithm. I thank Matteo Gallet for discussion and detailed suggestions on the motivation/background part of this paper, as well as some detailed modification suggestions.

References

[1] Matteo Gallet, Georg Grassegger, Josef Schicho. Counting realizations of Laman graphs on the sphere. *The Electronic Journal of Combinatorics*, Volume 27, Issue 2 (2020).

[2] Sean Keel. Intersection theory of moduli space of stable n-pointed vurves of genus zero. *Transaction of the American Mathematical Society*, **330** (1992), no. 2, 545-574.

Technical Reports of the Doctoral Program “Computational Mathematics”

2020

2020-01	N. Smoot: *A Single-Variable Proof of the Omega SPT Congruence Family Over Powers of 5* Feb 2020. Eds.: P. Paule, S. Radu
2020-02	A. Schafelner, P.S. Vassilevski: *Numerical Results for Adaptive (Negative Norm) Constrained First Order System Least Squares Formulations* March 2020. Eds.: U. Langer, V. Pillwein
2020-03	U. Langer, A. Schafelner: *Adaptive space-time finite element methods for non-autonomous parabolic problems with distributional sources* March 2020. Eds.: B. Jüttler, V. Pillwein
2020-04	A. Giust, B. Jüttler, A. Mantzaflaris: *Local (T)HB-spline projectors via restricted hierarchical spline fitting* March 2020. Eds.: U. Langer, V. Pillwein
2020-05	K. Banerjee, M. Ghosh Dastidar: *Hook Type Tableaux and Partition Identities* June 2020. Eds.: P. Paule, S. Radu
2020-06	A. Bostan, F. Chyzak, A. Jiménez-Pastor, P. Lairez: *The Sage Package* `comb_walks` *for Walks in the Quarter Plane* June 2020. Eds.: M. Kauers, V. Pillwein
2020-07	A. Meddah: *A stochastic multiscale mathematical model for low grade Glioma spread* June 2020. Eds.: E. Buckwar, V. Pillwein
2020-08	M. Ouafoudi: *A Mathematical Description for Taste Perception Using Stochastic Leaky Integrate-and-Fire Model* June 2020. Eds.: E. Buckwar, V. Pillwein
2020-09	A. Bostan, A. Jiménez-Pastor: *On the exponential generating function of labelled trees* July 2020. Eds.: M. Kauers, V. Pillwein
2020-10	J, Forcan, J. Qi: *How fast can Dominator win in the Maker-Breaker domination game?* July 2020. Eds.: V. Pillwein, J. Schicho
2020-11	J. Qi: *A calculus for monomials in Chow group of zero cycles in the moduli space of stable curves* Sept 2020. Eds.: P. Paule, J. Schicho

The complete list since 2009 can be found at
`https://www.dk-compmath.jku.at/publications/`